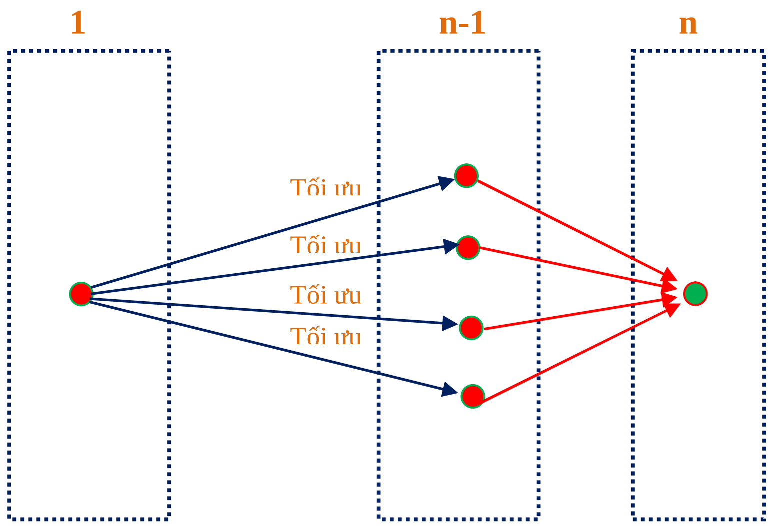
NGUYỄN KHẮC NHO

LÝ THUYẾT QUY HOẠCH ĐỘNG
VÀ ÁP DỤNG



NĂM 2012

Lời nói đầu

Trong các kỳ thi học sinh giỏi tin học như: Olympic 30/4; Học sinh giỏi quốc gia; Olympic tin học quốc tế…Thì các lớp bài toán về tối ưu hóa luôn được ưu tiên lựa chọn trong các đề thi, vì tính ứng dụng vào thực tiễn cao.

Có rất nhiều phương pháp để giải quyết lớp các bài toán tối ưu như: Phương pháp *nhánh cận*, phương pháp *tham lam*, phương pháp *quy hoạch động* (QHĐ)…Tùy từng bài toán cụ thể mà ta chọn 1 phương pháp để áp dụng nhằm đạt được hiệu quả (hiệu quả về phép tính toán (tốc độ), hiệu quả về bộ nhớ) tốt nhất. Trong đó phương pháp QHĐ luôn được ưu tiên lựa chọn vì tính hiệu quả của chúng cao hơn các phương pháp khác trong đại đa số các bài toán về tối ưu hóa.

Phương pháp QHĐ là một phương pháp khó bởi vì: Mỗi bài toán tối ưu có một đặc thù riêng, cách giải hoàn toàn khác nhau, cho nên cách áp dụng phương pháp QHĐ cho các bài toán cũng hoàn toàn khác nhau, không có khuân m ẫu chung cho tất cả các bài.

Phương pháp QHĐ là phương pháp giải quyết tốt các bài toán về tối ưu hóa nó cũng còn được áp dụng giải quyết một số bài toán không phải tối ưu và cũng đem lại hiệu quả cao. Việc xác định những bài toán như thế nào thì có thể áp dụng được phương pháp QHĐ vẫn còn rất khó khăn cho rất nhiều người.

Cho nên để giải quyết được nhiều bài toán khác nhau bằng PP QHĐ thì đòi hỏi người lập trình phải nắm vững bản chất của PP QHĐ. Với tài liệu này hi vọng sẽ giúp bạn đọc làm chủ được PP QHĐ một cách tự nhiên nhất.

Trong tài liệu này có sử dụng một số tài liệu tham khảo trên internet, một số cuốn sách chuyên tin, một số tài liệu trong và ngoài nước…

Xin chân thành cảm ơn

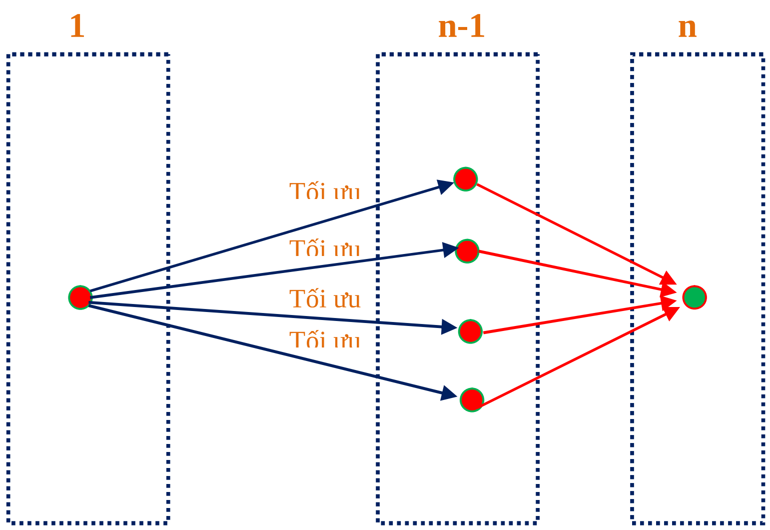
Mục Lục

CHƯƠNG 1: PHƯƠNG PHÁP QUY HOẠCH ĐỘNG

I. Khái niệm về phương pháp quy hoạch động (QHĐ)

Phương pháp quy hoạch động cùng nguyên lý tối ưu được nhà toán học Mỹ R.Bellman đề xuất vào những năm 50 của thế kỷ 20. Phương pháp này đã được áp dụng để giải hàng loạt bài toán thực tế trong các quá trình kỹ thuật cộng nghệ, tổ chức sản xuất, kế hoạch hoá kinh tế… Tuy nhiên cần lưu ý rằng có một số bài toán mà cách giải bằng quy hoạch động tỏ ra không thích hợp.

Nguyên lý tối ưu của R.Bellmam được phát biểu như sau: “*tối ưu bước thứ n bằng cách tối ưu tất cả con đường tiến đến bước n-1 và chọn con đường có tổng chi phí từ bước 1 đến bước n-1 và từ n-1 đến n là thấp nhất (nhiều nhất).*



Chú ý rằng nguyên lý này được thừa nhận mà không chứng minh.

Trong thực tế, ta thường gặp một số bài toán tối ưu loại sau: Có một đại lượng *f* hình thành trong một quá trình gồm nhiều giai đoạn và ta chỉ quan tâm đến kết quả cuối cùng là giá trị của *f* phải lớn nhất hoặc nhỏ nhất, ta gọi chung là giá trị tối ưu của *f*. Giá trị của *f* phụ thuộc vào những đại lượng xuất hiện trong bài toán mà mỗi bộ giá trị của chúng được gọi là một *trạng thái* của hệ thống và phụ thuộc vào cách thức đạt được giá trị *f* trong từng giai đoạn mà mỗi cách tổ chức được gọi là một *điều khiển*. Đại lượng *f* thường được gọi là *hàm mục tiêu* và quá trình đạt được giá trị tối ưu của *f* được gọi là *quá trình điều khiển tối ưu*.

Bellman phát biểu nguyên lý tối ưu (cũng gọi là *nguyên lý Bellman*) có thể diễn giải theo một cách khác như sau: “Với mỗi quá trình điều khiển tối ưu, đối với trạng thái bắt đầu A_0, với trạng thái *A* trong quá trình đó, phần quá trình kể từ trạng thái *A* xem như trạng thái bắt đầu cũng là tối ưu”.

Phương pháp tìm điều khiển tối ưu theo nguyên lý Bellman thường được gọi là *quy hoạch động*. Thuật ngữ này nói lên thực chất của quá trình điều khiển là động: có thể trong một số bước đầu tiên lựa chọn điều khiển tối ưu dường như không tốt nhưng tựu chung cả quá trình lại là tốt nhất.

Ta có thể giải thích ý này qua bài toán sau: Cho một dãy N số nguyên $A_1, A_2,\ldots,A_N$. Hãy tìm cách xoá đi một số ít nhất số hạng để dãy còn lại là đơn điệu hay nói cách khác hãy chọn một số nhiều nhất các số hạng sao cho dãy B gồm các số hạng đó theo trình tự xuất hiện trong dãy A là đơn điệu.

Quá trình chọn B được điều khiển qua N giai đoạn để đạt được mục tiêu là số lượng số hạng của dãy B là nhiều nhất, điều khiển ở giai đoạn i thể hiện việc chọn hay không chọn A_i vào dãy B.

Giả sử dãy đã cho là 1 8 10 2 4 6 7. Nếu ta chọn lần lượt 1, 8, 10 thì chỉ chọn được 3 số hạng nhưng nếu bỏ qua 8 và 10 thì ta chọn được 5 số hạng 1, 2, 4, 6, 7.

Khi giải một bài toán bằng cách "chia để trị" chuyển việc giải bài toán kích thước lớn về việc giải nhiều bài toán cùng kiểu có kích thước nhỏ hơn thì thuật toán này thường được thể hiện bằng các chương trình con đệ quy. Khi đó, trên thực tế, nhiều kết quả trung gian phải tính nhiều lần.

Vậy ý tưởng cơ bản của quy hoạch động thật đơn giản: tránh tính toán lại mọi thứ hai lần, mà lưu giữ kết quả đã tìm kiếm được vào một bảng làm giả thiết cho việc tìm kiếm những kết quả của trường hợp sau. Chúng ta sẽ làm đầy dần giá trị của bảng này bởi các kết quả của những trường hợp trước đã được giải. Kết quả cuối cùng chính là kết quả của bài toán cần giải. Nói cách khác phương pháp quy hoạch động đã thể hiện sức mạnh của nguyên lý chia để trị đến cao độ.

Quy hoạch động là kỹ thuật thiết kế bottom-up (từ dưới lên). Nó được bắt đầu với những trường hợp con nhỏ nhất (thường là đơn giải nhất và giải được ngay). Bằng cách tổ hợp các kết quả đã có (*không phải tính lại*) của các trường hợp con, sẽ đạt đạt tới kết quả của trường hợp có kích thước lớn dần lên và tổng quát hơn, cho đến khi cuối cùng đạt tới lời giải của trường hợp tổng quát nhất.

Trong một số trường hợp, khi giải một bài toán A, trước hết ta tìm họ bài toán A(p) phụ thuộc tham số p (có thể p là một véc tơ) mà $A(p_0)=A$ với p_0 là trạng thái ban đầu của bài toán A. Sau đó tìm cách giải họ bài toán A(p) với tham số p bằng cách áp dụng nguyên lý tối ưu của Bellman. Cuối cùng cho $p=p_0$ sẽ nhận được kết quả của bài toán A ban đầu.

II. Các bước thược hiện quy hoạch động

***Bước* 1: Lập hệ thức**

Dựa vào nguyên lý tối ưu tìm cách chia quá trình giải bài toán thành từng giai đoạn, sau đó tìm hệ thức biểu diễn tương quan quyết định của bước đang xử lý với các bước đã xử lý trước đó. Hoặc tìm cách phân rã bài toán thành các "bài toán con" tương tự có kích thước nhỏ hơn, tìm hệ thức nêu quan hệ giữa kết quả bài toán kích thước đã cho với kết quả của các "bài toán con" cùng kiểu có kích thước nhỏ hơn của nó nhằm xây dựng phương trình truy toán (dạng hàm hoặc thủ tục đệ quy).

Về một cách xây dựng phương trình truy toán:

Ta chia việc giải bài toán thành n giai đoạn. Mỗi giai đoạn i có trạng thái ban đầu là t(i) và chịu tác động điều khiển d(i) sẽ biến thành trạng thái tiếp theo t(i+1) của giai đoạn i+1 (i=1,2,…,n-1). Theo nguyên lý tối ưu của Bellman thì việc tối ưu giai đoạn cuối cùng không làm ảnh hưởng đến kết quả toàn bài toán. Với trạng thái ban đầu là t(n) sau khi làm giai đoạn n tốt nhất ta có trạng thái ban đầu của giai đoạn n-1 là t(n-1) và tác động điều khiển của giai đoạn n-1 là d(n-1), có thể tiếp tục xét đến giai đoạn n-1. Sau khi tối ưu giai đoạn n-1 ta lại có t(n-2) và d(n-2) và lại có thể tối ưu giai đoạn n-2 … cho đến khi các giai đoạn từ n giảm đến 1 được tối ưu thì coi như hoàn thành bài toán. Gọi giá trị tối ưu của bài toán tính đến giai đoạn k là F_k giá trị tối ưu của bài toán tính riêng ở giai đoạn k là G_k thì

$F_k = F_{k-1} + G_k$

Hay là: $F_1(t(k)) = \max_{\forall d(k)} \{G_k(t(k), d(k)) + F_{k-1}(t(k-1))\}$ (*)

***Bước* 2: Tổ chức dữ liệu và chương trình**

Tổ chức dữ liệu sao cho đạt các yêu cầu sau:

- Dữ liệu được tính toán dần theo các bước.
- Dữ liệu được lưu trữ để giảm lượng tính toán lặp lại.
- Kích thước miền nhớ dành cho lưu trữ dữ liệu càng nhỏ càng tốt, kiểu dữ liệu được chọn phù hợp, nên chọn đơn giản dễ truy cập.

Cụ thể

- Các giá trị của F_k thường được lưu trữ trong một bảng (mảng một chiều hoặc hai, ba, v.v… chiều).
- Cần lưu ý khởi trị các giá trị ban đầu của bảng cho thích hợp, đó là các kết quả của các bài toán con có kích cỡ nhỏ nhất của bài toán đang giải: $F_1(t(1)) = \max_{\forall d(1)} \{G_1(t(1), d(1)) + F_0(t(0))\}$
- Dựa vào công thức, phương trình truy toán (*) và các giá trị đã có trong bảng để tìm dần các giá trị còn lại của bảng.
- Ngoài ra còn cần mảng lưu trữ nghiệm tương ứng với các giá trị tối ưu trong từng gian đoạn.
- Dựa vào bảng lưu trữ nghiệm và bảng giá trị tối ưu trong từng giai đoạn đã xây dựng, tìm ra kết quả bài toán.

***Bước* 3: Làm tốt**

Làm tốt thuật toán bằng cách thu gọn hệ thức (*) và giảm kích thước miền nhớ. Thường tìm cách dùng mảng một chiều thay cho mảng hai chiều nếu giá trị một dòng (hoặc cột) của mảng hai chiều chỉ phụ thuộc một dòng (hoặc cột) kề trước.

Trong một số trường hợp có thể thay mảng hai chiều với các giá trị phần tử chỉ nhận giá trị 0, 1 bởi mảng hai chiều mới bằng cách dùng kỹ thuật quản lý bit.

III. Các thao tác tổng quát của phương pháp QHĐ

1. Xây dựng hàm QHĐ

2. Lập bảng lưu lại giá trị của hàm

3. Tính các giá trị ban đầu của bảng

4. Tính các giá trị còn lại theo kích thước tăng dần của bảng cho đến khi đạt được giá trị tối ưu cần tìm

5. Dùng bảng lưu để truy xuất lời giải tối ưu.

IV. Hạn chế của phương pháp QHĐ

Việc tìm công thức, phương trình truy toán hoặc tìm cách phân rã bài toán nhiều khi đòi hỏi sự phân tích tổng hợp rất công phu,dễ sai sót, khó nhận ra như thế nào là thích hợp, đòi hỏi nhiều thời gian suy nghĩ. Đồng thời không phải lúc nào kết hợp lời giải của các bài toán con cũng cho kết quả của bài toán lớn hơn.

Khi bảng lưu trữ đòi hỏi mảng hai, ba chiều … thì khó có thể xử lý dữ liệu với kích cỡ mỗi chiều lớn hàng trăm.

Có những bài toán không thể giải được bằng quy hoạch động.

CHƯƠNG 2: NHẬN DIỆN CÁC BÀI TOÁN CÓ THỂ GIẢI ĐƯỢC BẰNG PP QHĐ

I. Các bài toán không phải là bài toán tối ưu hóa.

Các bài toán có thể áp dụng được phương pháp QHĐ thì phải có tính chất: "***các bài toán con phủ chồng***".

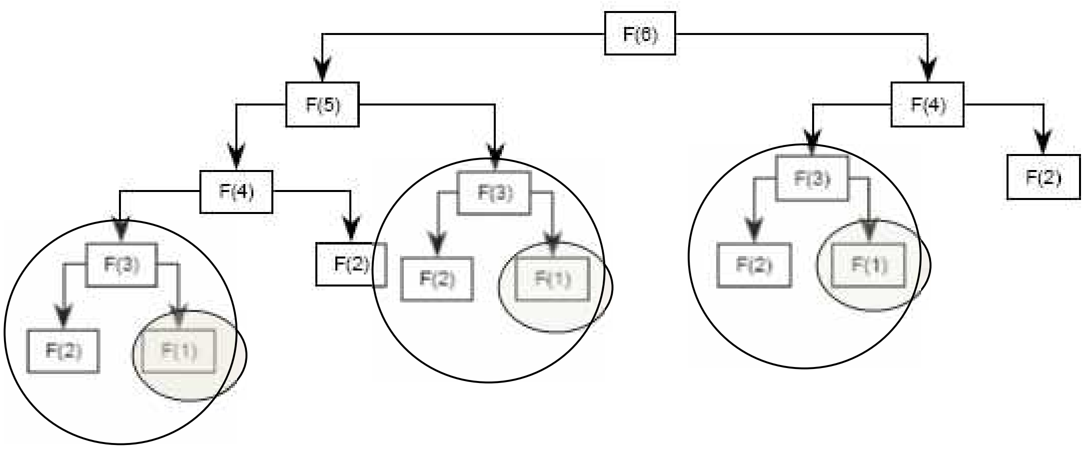
Có nghĩa là một bài toán có thể áp dụng phương pháp QHĐ thì một thuật toán đệ quy cho bài toán sẽ giải quyết lặp lại các bài toán con tương tự, thay vì luôn phát sinh bài toán con mới. Khi một thuật toán đệ quy ghé thăm hoài cùng một bài toán con, ta nói rằng bài toán có "các bài toán con phủ chồng". Ngược lại bài toán thích hợp với cách tiếp cận chia để trị thường phát sinh các bài toán con hoàn toàn mới tại mỗi bước đệ quy. Các thuật toán lập trình động thường vận dụng các bài toán phủ chồng bằng cách giải quyết từng bài toán con một rồi lưu trữ kết quả trong một bảng ở đó nó có thể được tra cứu khi cần.

<u>**Ví dụ 1**</u>: Bài toán tìm tính phần tử thứ n của dãy Fibonacci:

Dãy Fibonacci là dãy vô hạn các số tự nhiên bắt đầu bằng hai phần tử 0 và 1, các phần tử sau đó được thiết lập theo quy tắc *mỗi phần tử luôn bằng tổng hai phần tử trước nó*. Công thức truy hồi của dãy Fibonacci là:

$$F_n := F(n) := \begin{cases} 0, & \text{khi } n = 0; \\ 1, & \text{khi } n = 1; \\ F(n-1) + F(n-2) & \text{khi } n > 1. \end{cases}$$

Sơ đồ gọi đệ quy với bài toán tính F(6):



Ta thấy bài toán F(6) sẽ gọi bài toán F(5) và bài toán F(4). Cả bài toán F(5) và bài toán F(4) đều gọi bài toán F(3), hai bài toán con F(4) và F(3) lại cùng gọi bài toán con F(2)…ta gọi đó là tính chất "Bài toán con phủ chồng".

Một cài đặt đơn giản của một hàm tính phần tử thứ n của dãy Fibonacci, trực tiếp dựa theo định nghĩa toán học. Cài đặt này thực hi ện rất nhiều tính toán thừa.

function *fib(n)*

```
if n = 0 or n = 1
        return n
else
        return fib(n-1) + fib(n-2)
```

Lưu ý rằng nếu ta gọi, chẳng hạn: fib(5), ta sẽ tạo ra một cây các lời gọi hàm, trong đó các hàm của cùng một giá trị được gọi nhiều lần:

1. fib(5)
2. fib(4) + fib(3)
3. (fib(3) + fib(2)) + (fib(2) + fib(1))
4. ((fib(2) + fib(1)) + (fib(1) + fib(0))) + ((fib(1) + fib(0)) + fib(1))
5. (((fib(1) + fib(0)) + fib(1)) + (fib(1) + fib(0))) + ((fib(1) + fib(0)) + fib(1))

Áp dụng thuật toán QHĐ: Ta dùng một bảng để lưu trữ tất cả các bài toán con, như thế mỗi bài toán con chỉ phải tính một lần.

Cài đặt:

```
Begin
        A[0]:=0;
        A[1]:=1;
        For i:=2 to n do A[i]:=A[i-2] + A[i-1];
End.
```

Ví dụ 2: Bài Toán: **Mê Cung** (đề chọn đội tuyển tin Olympic 10 trường THPT chuyên Quang Trung năm 2010- 2011).

Phát biểu bài toán: Trong một chuyến thám hiểm mạo hiểm, một đoàn thám hiểm không may lọt vào một mê cung với nhiều cạm bẫy. Trong mê cung đó chỉ có một lối ra duy nhất, lối ra bao gồm các ô hình vuông được xếp thành một hàng dài. Muốn đi được ra ngoài mọi người phải bước qua một hàng các ô hình vuông đó và phải bước theo quy tắc sau:

- Quy tắc 1: Mỗi bước chỉ có thể bước một ô hoặc hai ô hoặc ba ô.
- Quy tắc 2: Từ người thứ 2 trở đi bước theo quy tắc 1 và không được trùng với các cách bước của tất cả những người trước đó.

Hỏi đoàn thám hiểm đó còn lại tối thiểu bao nhiêu người không thể thoát ra khỏi mê cung đó được.

Input Data:

- Dòng 1 ghi một số nguyên m ($m \leq 10^{18}$) là số người trong đoàn thám hiểm.
- Dòng 2 ghi một số nguyên n ($n \leq 70$) là tổng số ô vuông.

Output Data: Gồm 1 số nguyên duy nhất là số người còn lại tối thiểu không thể thoát ra khỏi mê cung

Ví dụ:

Input.inp	Output.out
20 5	7

Thực chất của bài toán là tìm xem có bao nhiêu cách bước ra ngoài.

Công thức truy hồi cho bài toán này được tính như sau:

+ Để bước lên ô thứ n chúng ta có 3 cách bước:

Cách 1: Bước tới ô thứ n-3 rồi bước 3 bước nữa

Cách 2: Bước tới ô thứ n-2 rồi bước 2 bước nữa

Cách 3: Bước tới ô thứ n-1 rồi bước 1 bước nữa

+ Theo nguyên lý cộng chúng ta có tổng số cách bước tới ô thứ n: **F(n) = F(n-3)+F(n-2) + F(n-1)** trong đó: F(1)= 1; F(2)= 2; F3(=4).

Chúng ta thấy bài toán xuất hiện rất nhiều "bài toán con phủ chồng" như bài toán tính phần tử thứ n của dãy Fibonacci. Chúng ta có thể áp dụng phương pháp QHĐ cho bài toán này để đạt được hiệu quả cao.

Mã nguồn:

```
Program Mecung;
Const   fi='';
        fo='';
Var     A:Array[1..100] of int64;
        i,n: byte;
        m: int64;
        f1,f2: text;
Begin
        Assign(f1,'fi');
        Reset (f1);
        Readln(f1,m);
        Read(f1,n);
        Close(f1);
        Assign(f2,'fo');
        Rewrite(f2);
        A[1]:=1;
        A[2]:=2;
        A[3]:=4;
        For i:=4 to n do A[i]:=A[i-1]+A[i-2]+A[i-3];
           If m> A[n] then
```

Write(f2, m-A[n]) else Write (f2,0);
Close (f2);
End.

II. Đối với các bài toán tối ưu:

Các bài toán tối ưu có thể áp dụng phương pháp QHĐ thì phải thỏa mãn 2 tính chất sau:

+ **Tính chất 1**: Các bài toán con phủ chồng

+ **Tính chất 2**: Cấu trúc con tối ưu: *Cấu trúc con tối ưu* có nghĩa là các lời giải tối ưu cho các bài toán con có thể được sử dụng để tìm các lời giải tối ưu cho bài toán toàn cục.

Ví dụ 1: Bài toán: Dãy con chung dài nhất:

Phát biểu bài toán: Xâu ký tự A được gọi là xâu con của xâu ký tự B nếu ta có thể xoá đi một số ký tự trong xâu B để được xâu A.

Cho biết hai xâu ký tự X và Y, hãy tìm xâu ký tự Z có độ dài lớn nhất và là con của cả X và Y.

Input

Dòng 1: chứa xâu X

Dòng 2: chứa xâu Y

Output: Chỉ gồm một dòng ghi độ dài xâu Z tìm được

Ví dụ:

Input.inp	Output.Out
ALGORITHM LOGARITHM	7

Ta đánh giá:

Tính chất 1: Ta có thể dễ thấy tính chất các bài toán con phủ chồng trong bài toán LCS (độ dài xâu con chung dài nhất). Để tìm một LCS của X và Y, có thể ta cần tìm các LCS của X và Y_{n-1} và của X_{m-1} và Y. Nhưng mỗi bài toán con này đều có bài toán cháu tìm LCS của X_{m-1} và Y_{n-1}. Nhiều bài toán con khác chia sẻ các bài toán cháu.

Tính chất 2: Ta có nhận xét như sau:

1. Nếu $x_m = y_n$, thì $z_k = x_m = y_n$ và Z_{k-1} là một LCS của X_{m-1} và Y_{n-1}.
2. Nếu $x_m \neq y_n$, thì $z_k \neq x_m$ hàm ý Z là một LCS của X_{m-1} và Y.
3. Nếu $x_m \neq y_n$, thì $z_k \neq y_n$ hàm ý Z là một LCS của X và Y_{n-1}.

Vậy ta hoàn toàn có thể tính được LCS của X và Y khi biết một LCS của X_{m-1} và Y_{n-1} hoặc LCS của X_{m-1} và Y hoặc LCS của X và Y_{n-1}.

Vậy bài toán thỏa mãn cả hai tính chất do vậy hoàn toàn có thể giải được bằng phương pháp QHĐ.

Hướng dẫn giải chi tiết:

Bước 1: Xác định đặc điểm của dãy con chung dài nhất (giải pháp tối ưu của bài toán)

Cho X = <x_1, x_2,…,x_m> và Y = <y_1, y_2,…,y_n> là các dãy, và Z = <z_1, z_2,…,z_k> là một dãy LCS bất kỳ của X và Y.

1. Nếu $x_m = y_n$, thì $z_k = x_m = y_n$ và Z_{k-1} là một LCS của X_{m-1} và Y_{n-1}.
2. Nếu $x_m \neq y_n$, thì $z_k \neq x_m$ hàm ý Z là một LCS của X_{m-1} và Y.
3. Nếu $x_m \neq y_n$, thì $z_k \neq y_n$ hàm ý Z là một LCS của X và Y_{n-1}.

Chứng minh:

1. Nếu $z_k \neq x_m$, thì ta có thể chắp $x_m = y_n$ vào Z để có được một dãy con chung của X và Y có chiều dài k+1, mâu thuẫn với giả thiết cho rằng Z là LCS của X và Y. Gả sử có một dãy con chung W của X_{m-1} và Y_{n-1} có chiều dài lớn hơn k-1. Như vậy việc chắp $x_m = y_n$ vào W sẽ tạo ra một dãy con chung của X và Y có chiều dài lớn hơn k, là một sự mâu thuẩn.
2. Nếu $z_k \neq x_m$, thì Z là một dãy con chung của X_{m-1} và Y. Nếu có một dãy con chung W của X_{m-1} và Y có chiều dài lớn hơn k, thì W cũng sẽ là sẽ là một dãy con chung của X và Y, mẫu thuẫn với giả thiết cho rằng Z là LCS của X và Y
3. Chứng minh đối xứng với (2)

Như vậy: Bài toán LCS có một tính chất cấu trúc con tối ưu (từ cấu trúc con tối ưu → dẫn đến bài toán tối ưu).

<u>Bước 2</u>: Một giải pháp đệ quy cho các bài toán con

Ta có thể dễ thấy tính chất các bài toán con phủ chồng trong bài toán LCS. Để tìm một LCS của X và Y, có thể ta cần tìm các LCS của X và Y_{n-1} và của X_{m-1} và Y. Nhưng mỗi bài toán con này đều có bài toán cháu tìm LCS của X_{m-1} và Y_{n-1}. Nhiều bài toán con khác chia sẻ các bài toán cháu.

Ta định nghĩa c[i,j] là chiều dài của một LCS của các dãy X_i và Y_j. Nếu i=0 hoặc j=0, một trong các dãy sẽ có chiều dài 0, do đó LCS có chiều dài 0. Cấu trúc con tối ưu của bài toán LCS cho công thức đệ quy:

$$c[i,j] = \begin{cases} 0 & if\ i = 0\ or\ j = 0 \\ c[i-1, j-1] + 1 & if\ i,j > 0\ and\ xi = yj \\ \max(c[i, j-1], c[i-1, j]) & if\ i,j > 0\ and\ xi \neq yi \end{cases}$$

<u>Bước 3</u>: Tính toán chiều dài của một LCS

Dựa trên công thức trên, ta có thể dễ dàng viết một thuật toán đệ quy thời gian mũ để tính toán chiều dài của một LCS cũa hai dãy. Tuy nhiên chỉ có O(mn) bài toán riêng biệt, nên ta có thể dùng lập trình động để tính toán các giải pháp từ dưới lên.

LCS – LENGTH (X, Y)

```
1.      m ← length (X)
2.      n ← length (Y)
3.      for i ← 1 to m
4.              do c[i,0] ← 0
5.      for j ← 1 to n
6.              do c[0,j] ← 0
7.      for i ← 1 to m
8.              do for j ← 1 to n
9.                      do if x_i =y_j
10.                     Then c[i,j] ← c[i-1, j-1]+1
11.                     B[i,j] ← "□"
12.             else if c[i-1,j] ≥ c[I,j-1]
13.             then c[i,i] ← c[i-1, j]
14.                     B[i,j] ← "↑"
15.             else c[i,i] ← c[i, j-1]
16.                     B[i,j] ← "←"
17. Return c và b
```

Thủ tục LCS – LENGTH tiếp nhận hai dãy X = <x_1, x_2,…,x_m> và Y = <y_1, y_2,…,y_n> làm nhập liệu. Nó lưu trữ c[i,j] giá trị trong một bảng c[0..m, 0..n] có các việc nhập được tính toán theo thứ tự chính là hàng (ghĩa là: Hàng đầu tiên của c được điền từ trái qua phải, sau đó đến hàng thứ 2…). Nó cũng duy trì bảng b[1..m, 1..n] để rút gọn việc kiến tạo một giải pháp tối ưu. Theo quan sát, b[i,j] trỏ đến việc nhập bảng tương ứng với giải pháp bài toán con tối ưu được chọn khi tính toán c[i,j]. Thủ tục trả về bảng b và c; c[m, n] chứa chiều dài của một LCS của X và Y.

Bước 4: Kiến tạo một LCS

		A	L	G	O	R	I	T	H	M
	0	0	0	0	0	0	0	0	0	0
L	0	0	1	1	1	1	1	1	1	1
O	0	0	1	1	2	2	2	2	2	2
G	0	0	1	2	2	2	2	2	2	2
A	0	1	1	2	2	2	2	2	2	2
R	0	1	1	2	2	3	3	3	3	3
I	0	1	1	2	2	3	4	4	4	4
T	0	1	1	2	2	3	4	5	5	5
H	0	1	1	2	2	3	4	5	6	6
M	0	1	1	2	2	3	4	5	6	7

Bảng b mà thủ tục LCS – LENGTH trả về có thể được dùng để nhanh chóng kiến tạo một LCS của X = <x_1, x_2,…,x_m> và Y = <y_1, y_2,…,y_n>. Ta đơn giản bắt đầu từ b[m,n] và rà qua bảng theo các mũi tên. Mỗi khi ta gặp một *“□” trong việc nhập b[i,j], nó hàm ý rằng x_i = y_j là một thành phần của LCS. Phương pháp này in ra một đề xuất đảo ngược. Thủ tục đệ quy sau đây in ra một LCS đúng đắn.*

PRINT – LCS (b,X,i, j)

1. if i =0 or j=0
2. Then reture
3. if b[i,j] = “□”
4. *Then Print – LCS (b,X,i-1,j-1);*
5. *Print x_i*
6. *else if b[i,j] = “↑”*
7. *Then Print – LCS (b,X,i-1,j);*
8. *else Print – LCS (b,X,i,j-1);*

Thủ tục PRINT – LCS (b,X,i, j) mất một thời gian O(m+n), bởi ít nhất một trong số biến i, j được giảm lượng trong mỗi giai đoạn đệ quy.

<u>Bước 5</u>: Cải thiện mã

Sau khi phát triển một thuật toán, bạn thường thấy có thể cải thiện thời gian hoặc không gian mà nó sử dụng. Điều này hoàn toàn dễ dàng thực hiện đối với các thuật toán lập trình động không phức tạp. Vài thay đổi có thể rút gọn mã và cải thiện các thừa số bất biến, các thay đổi có thể mang lại các khoản tiết kiệm đáng kể về thời gian và không gian.

Ví dụ: Có thể loại bỏ bảng b. Mỗi việc nhập c[i,j] chỉ tùy thuộc vào ba việc nhập bảng c: c[i-1, j-1], c[i-1,j], c[i, j-1] Cho giá trị c[i,j], ta có thể xác định trong O(1) thời gian giá trị nào trong 3 giá trị này được dùng để tính toán c[i,j], mà không kiểm tra bảng b. Như vậy, ta có thể dùng một thủ tục O(m +n) thời gian tương tự như thủ tục PRINT – LCS (b,X,i, j). Tuy tiết kiệm O(mn) không gian bằng phương pháp này song yêu cầu không gian phụ để để tính một LCS không giảm theo tiệm cận, bởi ta vẫn cần O(mn) không gian cho bảng c.

Tuy nhiên ta có thể rút gọn các yêu cầu không gian tiệm cận LCS – LENGTH, bởi nó chỉ cần hai hàng của bảng c vào một thời điểm: Hàng đang được tính toán và hàng trước đó. Áp dụng việc cải thiện kiểu này nếu ta chỉ cần tính chiều dài của LCS. Nếu cần kiến tạo lại các thành phần của một LCS, bảng nhỏ hơn không lưu giữ đủ thông tin để rà lại các bước của chúng ta trong O(m+n) thời gian.

Ví dụ 2: **Bài toán cái túi:**

Phát biểu bài toán: Cho N đồ vật, vật i có khối lượng W[i] và giá trị là V[i]. Một cái túi có thể chịu được khối lượng tối đa là M, quá thì sẽ rách. Hãy tìm cách nhét 1 số đồ vật vào trong túi sao cho túi không bị rách và tổng giá trị của các đồ vật nhét vào là lớn nhất.

Input

Dòng đầu tiên là số nguyên T là số bộ test. (1 ≤ T ≤ 40)

Mỗi bộ test sẽ có định dạng như sau:

+ Dòng 1: 2 số nguyên dương N, M (1 ≤ N ≤ 10000, 1 ≤ M ≤ 1000).

+ Dòng 2: Gồm N số nguyên là W[i] (1 ≤ W[i] ≤ 1000).
+ Dòng 3: Gồm N số nguyên là V[i] (1 ≤ V[i] ≤ 10000).

Output

Với mỗi bộ test:

+ Dòng đầu tiên ghi ra giá trị lớn nhất có thể đạt được và số K là số đồ vật lựa chọn.
+ Dòng thứ 2 ghi ra chỉ số của K đồ vật được chọn.

Input.inp	Output.Out
1 3 4 1 2 3 4 5 6	10 2 1 3

Ta đánh giá:

+ **Tính chất 1**: ***Bài toán con phủ chồng***: Khi ta xét một vật để xếp vào túi, ta phải so sánh giải pháp cho bài toán khi xếp vật được xếp vào túi với giải pháp cho bài toán con trước khi xếp vật đó vào cái túi. Bài toán được lập theo cách này gây ra nhiều bài toán con phủ chồng.

+ **Tính Chất 2**: ***Cấu trúc con tối ưu***: Ta hãy xét tải trọng có giá trị lớn nhất cân nặng tối đa W. Nếu gỡ bỏ mục j ra khỏi tải trọng này, tải trọng còn lại phải là tải trọng có giá trị nhất cân nặng tối đa W – w_j mà ta có thể xếp từ n-1 vật ban đầu trừ vật j.

CHƯƠNG 3: MỘT SỐ DẠNG ĐIỂN HÌNH CÁC BÀI TOÁN GIẢI BẰNG PP QHĐ

Trong chương này tôi chỉ trình bày cách cài đặt tường minh nhất về thuất toán QHĐ mà không đi sâu về cải tiến không gian và thời gian của chương trình. Khi các bạn nắm vững được nguyên lý của QHĐ thì rất dễ dàng trong việc phát hiện, phát triển và cải tiến thuật toán.

Một vài chú ý: Khi khai báo một biến kiểu kiểu thì mặc định biến đó bằng 0, kiểu boolean thì mặc định bằng false. Khai báo mảng các thành phần là kiểu số thì mặc định các phần tử bằng 0…Nên trong quá trình cài đặt một số bài toán sẽ không có câu lệnh để khởi tạo các giá trị như **fillchar(A,sizeof(A),0).**

Phương pháp QHĐ là một phương pháp khó bởi vì: Mỗi bài toán tối ưu có một đặc thù riêng, cách giải hoàn toàn khác nhau, cho nên cách áp dụng phương pháp QHĐ cho các bài toán cũng hoàn toàn khác nhau, không có khuân mẫu chung cho tất cả các bài. Khi các bạn đã nắng vững một số dạng toán quy hoạch động điểu hình thì việc phát hiện một bài toán giải bằng phương pháp QHĐ là trở lên dễ dàng.

BÀI 1: LỚP BÀI TOÁN CÁI TÚI

I. TỔNG QUAN

1. Mô hình

Trong siêu thị có n đồ vật (n≤1000), đồ vật thứ i có trọng lượng là W[i] ≤1000 và giá trị V[i] ≤1000. Một tên trộm đột nhập vào siêu thị, tên trộm mang theo một cái túi có thể mang được tối đa trọng lượng M (M≤1000). Hỏi tên trộm sẽ lấy đi những đồ vật nào để được tổng giá trị lớn nhất.

Giải quyết bài toán trong các trường hợp sau:

- Mỗi vật chỉ được chọn một lần.
- Mỗi vật được chọn nhiều lần (không hạn chế số lần)

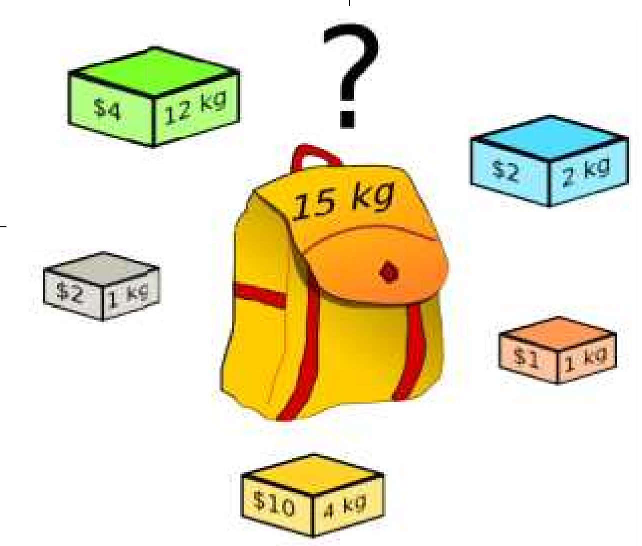
InputData: file văn bản **Bag.inp**

- Dòng 1: n, M cách nhau ít nhất một dấu cách
- n dòng tiếp theo: Mỗi dòng gồm 2 số V_i, W_i là chi phí và giá trị đồ vật thứ i.

OutputData: file văn bản **bag.out**: Ghi giá trị lớn nhất tên trộm có thể lấy

Example

Input	**Output**
5 15 12 4 2 2 1 1 1 2 4 10	15



2. Hướng dẫn giải

Trường hợp mỗi vật được chọn 1 lần

Ta nhận thấy rằng: Giá trị của cái túi phụ thuộc vào 2 yếu tố: Có bao nhiêu vật đang được xét và trọng lượng của các vật, do vậy chúng ta có 2 đại lượng biến thiên. Cho nên hàm mục tiêu sẽ phụ thuộc vào hai đại lượng biến thiên. Do vậy bảng phương án của chúng ta sẽ là bảng 2 chiều.

Gọi F[i,j] là tổng giá trị lớn nhất của cái túi khi xét từ vật 1 đến vật i và trọng của cái túi chưa vượt quá j. Với giới hạn j, việc chọn tối ưu trong số các vật {1,2,…,i-1,i} để có giá trị lớn nhất sẽ có hai khả năng:

Nếu không chọn vật thứ i thì F[i,j] là giá trị lớn nhất có thể chọn trong số các vật {1,2,…,i-1} với giới hạn trọng lượng là j, tức là:

F[i,j]:=F[i-1,j].

Nếu có chọn vật thứ i (phải thỏa điều kiện W[i] ≤ j) thì F[i,j] bằng giá trị vật thứ i là V[i] cộng với giá trị lớn nhất có thể có được bằng cách chọn trong số các vật {1,2,…,i-1} với giới hạn trọng lượng j-W[i] tức là về mặt giá trị thu được:

F[i,j]:=V[i]+F[i-1,j-W[i]]

Vậy chúng ta phải xem xét xem nếu chọn vật i hay không chọn vật i thì sẽ tốt hơn. Từ đó chúng ta có công thức truy hồi như sau.

- **F[0,j] = 0** (hiển nhiên) – Bài toán con nhỏ nhất.
- **F[i,j]= max(F[i-1,j], V[i]+F[i-1,j-W[i]]**

Trường hợp mỗi vật được chọn nhiều lần: Tương tự như suy luận ở trên ta xét:

Nếu không chọn vật thứ i thì F[i,j] là giá trị lớn nhất có thể chọn trong số các vật {1,2,…,i-1} với giới hạn trọng lượng là j, tức là:

F[i,j]:=F[i-1,j].

Nếu có chọn vật thứ i (phải thỏa điều kiện W[i] ≤ j) thì F[i,j] bằng giá trị vật thứ i là V[i] cộng với giá trị lớn nhất có thể có được bằng cách chọn trong số các vật {1,2,…,i} (vì vật i vẫn có thể được chọn tiếp) với giới hạn trọng lượng j-W[i] tức là về mặt giá trị thu được:

F[i,j]:=V[i]+F[i,j-W[i]]

Do vậy chúng ta có **công thức truy hồi** như sau:

- **F[0,j] = 0** (hiển nhiên) – Bài toán con nhỏ nhất.
- **F[i,j]= max(F[i-1,j], V[i]+F[i,j-W[i]]**

3. Bảng phương án

Ta xây dựng bảng phương án dựa trên công thức truy hồi trên. Để kiểm tra kết quả có chính xác hay không (nếu không chính xác chúng ta xây dựng lại hàm mục tiêu). Thông qua cách xây dựng hàm mục tiêu và bảng phương án chúng ta sẽ định hướng việc truy vết.

Example (trường hợp mỗi vật chỉ được chọn 1 lần)

Input	**Output**
5 15 12 4 2 2 1 1 1 2 4 10	15

Bảng phương án:

N/M	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	4	4	4	4
2	0	0	**2**	2	2	2	2	2	2	2	2	2	4	4	6	6
3	0	1	2	**3**	3	3	3	3	3	3	3	3	4	5	6	7
4	0	2	3	4	**5**	5	5	5	5	5	5	5	5	6	7	8
5	0	2	3	4	10	12	13	14	**15**	15	15	15	15	15	15	15

Vậy chúng ta có thể chọn vật 2, 3, 4, 5.

Example (trường hợp mỗi vật được chọn nhiều lần)

Input	**Output**
5 15 12 4 2 2 1 1 1 2 4 10	36

Bảng phương án:

N/M	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	4	4	4	4
2	0	0	2	2	4	4	6	6	8	8	10	10	12	12	14	14
3	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
4	0	2	4	6	8	10	12	14	16	18	20	22	24	26	28	30
5	0	2	4	6	10	12	14	16	20	22	24	26	30	32	34	36

Chúng ta có thể chọn vật 4 (3 lần) và vật 5 (3 lần).

4. Truy vết

Trường hợp 1: Trong bảng phương án F[n,m] chính là giá trị lớn nhất thu được khi chọn trong cả n vật với giới hạn trọng lượng là M.

Nếu f[n,M]=f[n-1,M] thì tức là không chọn vật thứ n, ta truy về f[n-1,M]. Còn nếu f[n,M]≠f[n-1,M] thì ta thông báo rằng phép chọn tối ưu có chọn vật thứ n và truy về f[n-1,M-Wn].

Trường hợp 2: Trong bảng phương án F[n,m] chính là giá trị lớn nhất thu được khi chọn trong cả n vật với giới hạn trọng lượng là M.

Nếu f[n,M]=f[n-1,M] thì tức là không chọn vật thứ n, ta truy về f[n-1,M]. Còn nếu f[n,M]≠f[n-1,M] thì ta thông báo rằng phép chọn tối ưu có chọn vật thứ n và truy về f[n,M-Wn].

5. Cài đặt

```
Program Bag;
Const
        limit = 1000;
Var
        V,W: Array[1..limit] of integer;
        F: Array[0..limit,0..limit] of integer;
        n,M,i,j: integer;
        fi,fo: text;
{----------------------------------------------------------------------}
Procedure inputdata;
        Begin
                assign(fi,'bag.inp');
                reset(fi);
                readln(fi,n,m);
                for i:=1 to n do readln(fi,w[i],v[i]);
                close(fi);
        End;
{----------------------------------------------------------------------}
Procedure outputdata;
        Begin
                assign(fo,'bag.out');
                rewrite(fo);
                write(fo,F[n,m]);
                close(fo);
        End;
{----------------------------------------------------------------------}
Function max(a,b:integer):integer;
        Begin
                if a>b then max:=a
                else max:=b;
        End;
{----------------------------------------------------------------------}
Procedure process;
        Begin
                for i:=1 to n do
```

```
                    for j:=1 to m do
                        if w[i]<=j then
                            F[i,j]:=max(F[i-1,j],F[i-1,j-w[i]]+v[i])
                        else    F[i,j]:=F[i-1,j];
        End;
{-----------------------------------------------------------------------}
Begin
        inputdata;
        process;
        outputdata;
End.
```

II. Áp dụng

Bài toán 1: Farmer - Người nông dân (IOI 2004)

a. Phát biểu bài toán:

Một người nông dân có một số các cánh đồng, mỗi một cánh đồng được bao quanh bởi các hàng cây bách. Ngoài ra ông ta còn có một tập các dải đất, mỗi một dải đất có một hàng cây bách. Trên các cánh đồng và dải đất, xen giữa hai cây bách liên tiếp là một cây ôliu. Tất cả các cây bách hoặc bao quanh cánh đồng hoặc nằm trên dải đất và tất cả các cây ôliu đều được trồng xen giữa hai cây bách liên tiếp.

Một ngày nọ người nông dân bị ốm rất nặng và ông ta cảm thấy mính sắp phải đi xa. Vài ngày trước khi qua đời ông đã gọi người con trai lớn nhất đến và nói với anh ta "Ta cho con chọn Q cây bách bất kỳ và tất cả các cây ôliu nằm giữa hai cây bách liên tiếp mà con đã chọn đều thuộc về con". Người con có thể chọn tổ hợp các cây bách bất kỳ từ các cánh đồng và dải đất. Vì người con rất thích ôliu nên anh ta muốn chọn Q cây bách sao cho anh ta thừa hướng nhiều cây ôliu nhất có thể.

Trong hình dưới, giả thiết rằng người con được cho 17 cây bách (Q=17). Để có được số cây ôliu lớn nhất anh ta phải chọn tất cả các cây bách trong cánh đồng 1 và cánh đồng 2, với cách chọn này anh ta sẽ được thừa hưởng 17 cây ôliu.

Cho trước thông tin về các cánh đồng và dải đất và số cây bách người con được chọn. Bạn hãy viết chương trình xác định số cây ôliu lớn nhất mà người con có thể được thừa hưởng.

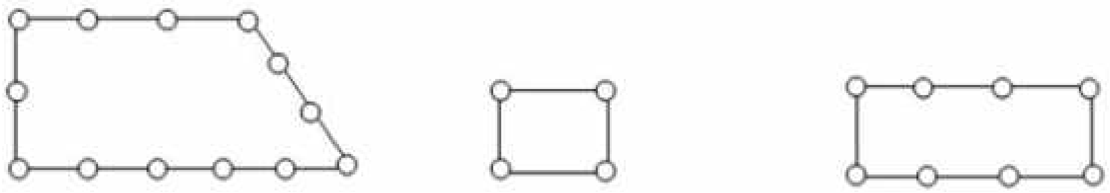
Inputdata: Dữ liệu được cho trong file **Farmer.in**

Dòng đầu tiên bao gồm: đầu tiên là số nguyên Q ($0 \le Q \le 150000$): là số cây bách mà người con được chọn; sau đó là số nguyên M là số các cánh đồng; tiếp theo là số nguyên K là số dải đất.

- Dòng thứ hai chứa M số nguyên $N_1, N_2, \ldots, N_M$ ($3 \le N_1, N_2, \ldots, N_M \le 150$): là số các cây bách trên các cánh đồng.
- Dòng thứ ba chứa K số nguyên $R_1, R_2, \ldots, R_K$ ($2 \le R_1, R_2, \ldots, R_K \le 150$): là số cây bách trên dải đất.

Chú ý: *tổng số cây bách trên các cánh đồng và dải đất ít nhất cũng bằng Q*

Outputdata: Kết quả đưa ra tệp **Farmer.out**: Gồm một duy nhất một số nguyên: Là số cây ôliu lớn nhất mà người con có thể thừa hưởng.



Cánh đồng 1 có 13 cây bách   Cánh đồng 2 có 4 cây bách   Cánh đồng 3 có 8 cây bách

Dải đất 1 có 4 cây bách

Dải đất 2 có 8 cây bách

Dải đất 3 có 6 cây bách

Hình trên: Ví dụ về cách bố trí các cây bách

Example:

Farmer.inp	**Farmer.out**
17 3 3 13 4 8 4 8 6	**17**

b. Hướng dẫn giải

Dễ thấy rằng: Mảnh đất thứ i có a_i cây ôliu và giải đất thứ j có b_j-1 cây ôliu. Coi các mảnh đất và các giải đất là các "đồ vật", đồ vật thứ k có khối lượng w_k (số cây bách) và giá trị v_k (số cây ôliu). Nếu k là mảnh đất i thì w_k=v_k=a_i, nếu k là dải đất thì w_k=b_i, v_k=b_j-1). Ta cần tìm các đồ vật sao cho tổng "khối lượng" của chúng không vượt quá Q và tổng "giá trị" là lớn nhất. Đây chính là bài toán cái túi (trường hợp thứ nhất) đã trình bày ở trên.

c. Cài đặt

```
Program farmer;
Const
        limit = 1000;
Var
        W,V: Array[1..2*limit] of byte;
        F:Array[0..2*limit,0..limit] of word;
        m,n,k,q,i,j:word;
        fi,fo:text;
{--------------------------------------------------------}
Procedure inputdata;
```

```
        Begin
            assign(fi,'farmer.inp');
            reset(fi);
            readln(fi,Q,M,N);
            for i:=1 to m do
                begin
                    read(fi,W[i]);
                    V[i]:=W[i];
                end;
            for i:=m+1 to n+m do
                begin
                    read(fi,W[i]);
                    V[i]:=W[i]-1;
                end;
            close(fi);
       End;
{-------------------------------------------------------}
Procedure outputdata;
        Begin
            assign(fo,'farmer.out');
            rewrite(fo);
            write(fo,F[M+N,Q]);
            close(fo);
        End;
{-------------------------------------------------------}
Function max(x,y:longint):longint;
        Begin
            if x>y then max:=x else max:=y;
        End;
{-------------------------------------------------------}
Procedure process;
        Begin
            for i:=1 to m+n do
                for j:=1 to q do
                    if j-W[i]>=0 then
                        F[i,j]:=max(F[i-1,j-W[i]]+V[i],F[i-1,j])
                    else F[i,j]:=F[i-1,j];
        End;
{-------------------------------------------------------}
Begin
        inputdata;
        process;
        outputdata;
End.
```

Bài toán 2: Đổi tiền

a. Mô hình

Bạn được cho một tập hợp các mệnh giá tiền. Tập hợp luôn chứa phần tử mang gía trị 1. Mỗi mệnh giá có vô hạn các đồng tiền mang mệnh giá đó. Cho số tiền S, hãy tìm cách đổi S thành ít đồng tiền nhất, sao cho mỗi đồng tiền có mệnh giá thuộc vào tập hợp đã cho.

Inputdata: file văn bản **doitien.inp** Dữ liệu vào gồm 2 dòng **:**

- Dòng 1: Hai số nguyên dương N (số phần tử của tập hợp mệnh giá tiền) và S (số tiền cần đổi) ($1 \le N \le 100$; $1 \le S \le 10^9$).
- Dòng 2: N số nguyên dương biểu thị mệnh giá của các phần tử trong tập hợp (giá trị không vượt quá 100).

Outputdata: file văn bản **doitien.out**: Gồm một số nguyên duy nhất là số đồng tiền ít nhất có thể đổi được.

Example

Doitien.inp	Doitien.out
2 3 1 2	2

b. Hướng dẫn giải

Ta nhận thấy: Nếu coi "khối lượng" là mệnh giá, "giá trị" là 1 thì đây chính là bài toán cái túi (trường hợp 2) ở trên. Chỉ có điểm khác là yêu cầu tính tổng giá trị nhỏ nhất.

Do đó ta xây dựng hàm mục tiêu một cách tương tự: Gọi F[i,t] là số đồng xu ít nhất nếu đổi t đồng ra i loại tiền xu (từ 1 đến i). Công thức tính F[i,t] như sau:

- F[i,0]=0;
- F[0,t]=∞ với t>0;
- F[i,t]=F[i-1,t] nếu t< wi
- F[i,t]:=min(F[i-1,t],F[i,t-wi]+1) nếu t ≥ wi

c. Cài đặt

```
Program DoiTien;
Program DoiTien;
Const
        limit = 100;
        vocung=200;
Var
        W: Array[1..limit] of byte;
        F: Array[0..limit,0..1000000] of integer;
        n,S,i,j: integer;
        fi,fo: text;
{--------------------------------------------------------------------}
Procedure inputdata;
```

```
    Begin
        assign(fi, ‘ DoiTien.inp’);
        reset(fi);
        readln(fi,n,S);
      for i:=1 to n do read(fi,w[i]);
      close(fi);
    End;
{------------------------------------------------------------------------}
Procedure outputdata;
    Begin
        assign(fo,'DoiTien.out');
        rewrite(fo);
        writeln(fo,F[n,S]);
        close(fo);
    End;
{------------------------------------------------------------------------}
Function min(a,b:integer):integer;
    Begin
        if a<b then min:=a
        else min:=b;
    End;
{------------------------------------------------------------------------}
Procedure process;
    Begin
      for j:=1 to s do F[0,j]:=vocung;
        for i:=1 to n do
            for j:=1 to S do
                if j-w[i]>=0 then
                    F[i,j]:=min(F[i-1,j],F[i,j-w[i]]+1)
                else    F[i,j]:=F[i-1,j];
    End;
{------------------------------------------------------------------------}
Begin
    inputdata;
    process;
    outputdata;
End.
```

BÀI 2: LỚP BÀI TOÁN DÃY CON ĐƠN ĐIỆU DÀI NHẤT

I. Tổng quan

1. Mô hình

Cho một dãy số nguyên gồm N phần tử A[1], A[2], ... A[N]. Biết rằng dãy con tăng đơn điệu là 1 dãy A[i_1],... A[i_k] thỏa mãn $i_1 < i_2 < ... < i_k$ và A[i_1] < A[i_2] < .. < A[i_k]. Hãy cho biết dãy con tăng đơn điệu dài nhất của dãy này có bao nhiêu phần tử?

Inputdata: File văn bản **liq.inp**

- Dòng 1 gồm 1 số nguyên là số N (1 ≤ N ≤ 1000).
- Dòng thứ 2 ghi N số nguyên A[1], A[2], .. A[N] (1 ≤ A[i] ≤ 10000).

Outputdata: File văn bản **lip.out**: Ghi ra độ dài của dãy con tăng đơn điệu dài nhất.

Example

liq.inp	Liq.out
6 1 2 5 4 6 2	4

Giải thích test ví dụ: Dãy con dài nhất là dãy A[1] = 1 < A[2] = 2 < A[4] = 4 < A[5] = 6, độ dài dãy này là 4.

2. Hướng dẫn giải

Vì các phần tử trong dãy kết quả chỉ xuất hiện đúng một lần nên ta sẽ cho duyệt qua tất cả các phần tử từ phần tử đầu tiên đến phần tử cuối cùng.

Tại mỗi phần tử a_i ta tính xem độ dài dãy con đơn điệu tăng dài nhất từ phần tử a_1 đến phần tử a_i là bao nhiêu. Do đó hàm mục tiêu chỉ phụ thuộc vào một đại lượng biến thiên nên ta có thể dùng bảng 1 chiều để lưu trữ bảng phương án.

Gọi F[i] là độ dài dãy con tăng dài nhất, các phần tử lấy trong miền từ a_1 đến a_i và phần tử cuối cùng là a_i. Dãy con đơn điêu tăng dài nhất kết thúc tại a_i sẽ được thành lập bằng cách:

- Kiểm tra xem có bao nhiêu dãy con đơn điệu tăng (các phần tử trong miền từ a_1 đến a_{i-1}) mà ta có thể gép ai vào cuối các dãy đó.
- Chọn dãy con đơn điệu có độ dài lớn nhất để ghép ai vào.

Với những suy luận ở trên ta sẽ xây dựng hàm mục tiêu như sau:

- F[1]:=1 (hiển nhiên)
- F[i]=max(1,F[j]+1 với mọi phần tử j: 0<j<i và a_j<a_i)

Tính F[i]: Phần tử đang được xét là a_i. Ta tìm đến phần tử a_j<a_i có F[j] lớn nhất. Khi đó nếu bổ xung a_i vào sau dãy con …a_j ta sẽ đượ dãy con tăng dài nhất xét từ a_1..a_i

Vậy kết quả của bài toán là F[i] nào lớn nhất trong số các F[i]: i:=1..n.

3. Bảng phương án

Ta xây dựng bảng phương án dựa trên công thức truy hồi trên. Để kiểm tra kết quả có chính xác hay không (nếu không chính xác chúng ta xây dựng lại hàm mục tiêu). Thông qua cách xây dựng hàm mục tiêu và bảng phương án chúng ta sẽ định hướng việc truy vết.

i	*1*	*2*	*3*	*4*	*5*	*6*
ai	**1**	**2**	**5**	**4**	**6**	**2**
F[i]	1	2	3	3	**4**	2

Vậy dãy con đơn điệu có độ dài lớn nhất là 4 và phần tử cuối cùng là a_5=6.

4. Cài đặt

```
Program LIQ;
Var
        A,F: Array[0..1000] of word;
        i,j,max,n:word;
        fi,fo: text;
{-----------------------------------------------------------------}
Procedure inputdata;
        Begin
                assign(fi,'liq.inp');
                reset(fi);
                readln(fi,n);
                for i:=1 to n do read(fi,a[i]);
                close(fi);
        End;
{-----------------------------------------------------------------}
Procedure outputdata;
        Begin
                assign(fo,'liq.out');
                rewrite(fo);
                write(fo,max);
                close(fo);
        End;
{-----------------------------------------------------------------}
Procedure process;
        Begin
                for i:=1 to n do
                        begin
```

```
                f[i]:=1{tối thiểu dãy cũng có 1 phần tử là ai};
                for j:=1 to i-1 do if (a[i]>a[j]) and (f[i]<f[j]+1)
                                        then f[i]:=f[j]+1;
                if max<f[i] then max:=f[i];
            end;
    End;
{------------------------------------------------------------------}
Begin
    inputdata;
    process;
    outputdata;
End.
```

II. Áp dụng

Bài toán 1: Bố trí phòng họp

a. Phát biểu bài toán

Có n cuộc họp, cuộc họp thứ i bắt đầu vào thời điểm ai và kết thúc ở thời điểm bi. Do chỉ có một phòng hội thảo nên 2 cuộc họp bất kì sẽ được cùng bố trí phục vụ nếu khoảng thời gian làm việc của chúng chỉ giao nhau tại đầu mút. Hãy bố trí phòng họp để phục vụ được nhiều cuộc họp nhất.

Inputdata: file văn bản **BTPH.inp**:

- Dòng 1: Số nguyên dương N (1≤N≤1000).
- N dòng tiếp theo: Dòng thứ i chứa chứa hai số nguyên dương (≤1000) chỉ thời điểm bắt đầu và thời điểm kết thúc của cuộc họp thứ i.

Outputdata: file văn bản **BTPH.out**: Một số duy nhất là số cuộc họp nhiều nhất.

Example

BTPH.inp	BTPH.out
4 4 5 5 6 1 6 6 9	3

b. Hướng dẫn giải

Sắp xếp các cuộc họp tăng dần theo thời điểm bắt đầu (ai). Thế thì cuộc họp i sẽ bố trí được sau cuộc họp j nếu và chỉ nếu j<i và $b_j \le$ ai. Yêu cầu bố trí được nhiều cuộc họp dài nhất có thể đưa về việc tìm dãy các cuộc họp dài nhất thỏa mãn điều kiện trên.

c. Cài đặt

```
Program BoTriPhongHop;
Type Gio=record
                gbd: word;{giờ bắt đầu}
                gkt: word;{giờ kết thúc}
                End;
Var
        A: Array[1..1000] of Gio;
        F: Array[0..1000] of word;
        i,j,n,max: word;
        fi,fo:text;
{-------------------------------------------------------------}
Procedure inputdata;
        Begin
                assign(fi,'BTPH.inp');
                reset(fi);
                readln(fi,n);
                for i:=1 to n do readln(fi,A[i].gbd,A[i].gkt);
                close(fi);
        End;
{-------------------------------------------------------------}
Procedure outputdata;
        Begin
                assign(fo,'BTPH.out');
                rewrite(fo);
                write(fo,max);
                close(fo);
        End;
{-------------------------------------------------------------}
Procedure sort;
        var tam: Gio;
        Begin
                for i:= 1 to n-1 do
                        for j:=i+1 to n do
                                if a[i].gbd > a[j].gbd then
                                        begin
                                                tam:=a[i];
                                                a[i]:=a[j];
                                                a[j]:=tam;
                                        end;
        End;
```

```
{--------------------------------------------------------------}
Procedure process;
     Begin
          max:=0;
          for i:=1 to n do
               begin
                    f[i]:=1;
                    for j:=1 to n-1 do
                         if (a[i].gbd >= a[j].gkt) and (f[i]<f[j]+1)
                              then f[i]:=f[j]+1;
                    if f[i]>max then max:=f[i];
               end;
     End;
{--------------------------------------------------------------}
Begin
     inputdata;
     sort;
     process;
     outputdata;
End.
```

Bài toán 2: Cho Thuê máy

a. Phát biểu bài toán

Trung tâm tính toán hiệu năng nhận được đơn đặt hàng của N khách hàng. Khách hàng i muốn thuê máy từ ngày a_i đến ngày b_i và trả c_i tiền thuê. Hãy bố trí lịch thuê máy để tổng số tiền thu được là lớn nhất mà thời gian sử dụng máy của 2 khách hàng bất kì không được giao nhau (trung tâm chỉ có một máy cho thuê).

Inputdata: file văn bản **thuemay.inp**

- Dòng đầu tiên là số N (N≤10000).
- N dòng tiếp theo dòng thứ i ghi 3 số a_i , b_i , c_i ($1 \le a_i$,$b_i \le 100$)

Outputdata: file văn bản **thuemay.out**: Một dòng duy nhất là tổng số tiền lớn nhất thu được.

Example

THUEMAY.INP	THUEMAY.OUT
3 1 8 16 2 7 6 7 9 9	16

b. Hướng dẫn giải

Tương tự như bài toán bố trí phòng họp. Sắp xếp các đơn đặt hàng theo thời điểm bắt đầu. Đây là bài toán biến thể của bài toán tìm dãy con tăng dài nhất. Bài toán này là tìm dãy con có tổng lớn nhất.

c. Cài đặt

```
Program thuemay;
Type Gio=record
            gbd: word; {gio bat dau}
            gkt: word; {gio ket thue}
            tt :word;{tien thue may}
            End;
Var
      A: Array[1..1000] of Gio;
      F: Array[0..1000] of longint;
      i,j,n,max: word;
      fi,fo:text;
{-------------------------------------------------------------}
Procedure inputdata;
      Begin
            assign(fi,'thuemay.inp');
            reset(fi);
            readln(fi,n);
            for i:=1 to n do readln(fi,A[i].gbd,A[i].gkt,A[i].tt);
            close(fi);
      End;
{-------------------------------------------------------------}
Procedure outputdata;
      Begin
            assign(fo,'thuemay.out');
            rewrite(fo);
            writeln(fo,max);
         for i:=1 to n do writeln(fo,a[i].gbd,' ',a[i].gkt,' ',a[i].tt,' ',f[i]);
            close(fo);
      End;
{-------------------------------------------------------------}
Procedure sort;
      var tam: Gio;
      Begin
            for i:= 1 to n-1 do
```

```
                    for j:=i+1 to n do
                        if a[i].gbd > a[j].gbd then
                            begin
                                tam:=a[i];
                                a[i]:=a[j];
                                a[j]:=tam;
                            end;
        End;
{-------------------------------------------------------------}
Procedure process;
        Begin
            max:=0;
            for i:=1 to n do
                begin
                    f[i]:=a[i].tt;
                    for j:=1 to n-1 do
                        if (a[i].gbd >= a[j].gkt) and (f[i]<f[j]+a[i].tt)
                            then f[i]:=f[j]+a[i].tt;
                    if f[i]>max then max:=f[i];
                end;
        End;
{-------------------------------------------------------------}
Begin
        inputdata;
        sort;
        process;
        outputdata;
End.
```

Bài toán 3: Dãy đổi dấu

a. Phát biểu bài toán

Cho dãy a1, a2,…an. Hãy dãy con đổi dấu dài nhất của dãy đó. Dãy con con đổi dấu ai1,ai2,…aik phải thoả mãn các điều kiện sau:

- ai1<ai2>ai3<… hoặc ai1>ai2<ai3>…
- Các chỉ số phải cách nhau ít nhất L: i2–i1≥L, i3–i2≥L….
- Chênh lệch giữa 2 phần tử liên tiếp nhỏ hơn U: |ai1–ai2|≤U, |ai2–ai3|≤U…

Inputdata: file văn bản **daydoidau.inp**:

- Dòng đầu gồm 3 số nguyên N,L,U (1<L<L≤10000, U≤10000);
- Dòng thứ 2 là n số nguyên ai (|ai|≤10000);

Outputdata: file văn bản **daydoidau.out**: Độ dài dãy đổi dấu dài nhất.

Example

Daydoidau.inp	Daydoidau.out
5 2 2 5 4 3 2 7	3

Giải thích: Ta có dãy đổi dấu là 5 3 7

b. Hướng dẫn giải

Gọi F[i] là số phần tử của dãy con đổi dấu có phần tử cuối cùng là ai và phần tử cuối cùng lớn hơn phần tử đứng trước. Gọi P[i] là số phần tử của dãy con đổi dấu có phần tử cuối cùng là ai và phần tử cuối cùng nhỏ hơn phần tử đứng trước.

Ta dễ dàng suy ra:

- F[i]=max(1,P[j]+1): j≤i-L và $a_i - a_j \le U$
- P[i] = max(1,F[i]+1): j≤i-L và a_j-$a_i \le U$

c. Cài đặt

```
Program DayDoiDau;
Const
        limit = 10000;
Var
        A,F,Q: Array[1..limit] of longint;
        i,j,n,l,u,maximum: longint;
        fi,fo:text;
{---------------------------------------------------------------------}
Function max(a,b:longint):longint;
        Begin
                if a>b then max:=a else max:=b;
        End;
{---------------------------------------------------------------------}
Procedure inputdata;
        Begin
                assign(fi,'daydoidau.inp');
                reset(fi);
                readln(fi,n,l,u);
                for i:=1 to n do read(fi,a[i]);
                close(fi);
        End;
{---------------------------------------------------------------------}
```

```
Procedure outputdata;
        Begin
                assign(fo,'daydoidau.out');
                rewrite(fo);
                write(fo,maximum);
                close(fo);
        End;
{----------------------------------------------------------------------}
Procedure process;
        Begin;
    f[1]:=1; q[1]:=1;
    maximum:=f[1];
                for i:=2 to n do
                        begin
                                f[i]:=1; q[i]:=1;
                                for j:=1 to i-1 do
                                        begin
                                                if(a[i]>a[j])and(i-j>=l)and(a[i]-
                                                a[j]<=u)and(f[i]<q[j]+1)
                                                        then f[i]:=q[j]+1;
                                                if(a[j]>a[i])and(i-j>=l)and(a[j]-
                                                a[i]<=u)and(q[i]<f[j]+1)
                                                        then q[i]:=f[j]+1;
                                        end;
                                if(maximum<f[i])or(maximum<q[i])then
                                maximum:=max(f[i],q[i]);
                        end;
        end;
{----------------------------------------------------------------------}
Begin
        inputdata;
        process;
        outputdata;
End.
```

Bài toán 4: Dãy Wavio

a. Phát biểu bài toán

Dãy số Wavio là dãy số nguyên thỏa mãn các tính chấ t: Các phần tử đầu sắp xếp thành 1 dãy tăng dần đến 1 phần tử đỉnh sau đó giảm dần. Ví dụ dãy số 1 2 3 4 5 2 1 là 1 dãy Wavio độ dài 7. Cho 1 dãy gồm N số nguyên, hãy chỉ ra một dãy con Wavio có độ dài lớn nhất trích ra từ dãy đó.

Inputdata: file văn bản **WAVIO.INP**:

- Dòng 1 : 1 số nguyên n duy nhất (n ≤10000)
- Dòng 2 : n số nguyên, các số cách nhau bằng 1 dấu cách (≤10000).

Outputdata: file văn bản **WAVIO.OUT**:1 số nguyên duy nhất là độ dài dãy WAVIO dài nhất tìm được.

Example

WAVIO.INP	WAVIO.OUT
5 2 1 4 3 5	3

b. Hướng dẫn giải

Gọi F1[i] là mảng ghi độ dài lớn nhất của 1 dãy con tăng trích ra từ dãy n phần tử ban đầu kể từ phần tử a_1 đến phần tử a_i. F2[i] là mảng ghi độ dài lớn nhất của dãy con giảm dần trích ra từ dãy N phần tử kể từ phần tử kể từ phần tử a_n đến a_i. Tìm phần tử j trong 2 mảng F1 và F2 thỏa mãn F1[j]+F2[j] lớn nhất.

c. Bảng phương án

i	1	2	3	4	5
ai	**2**	**1**	**4**	**3**	**5**
F1	1	1	2	2	3
F2	3	3	2	2	1

Vậy F1+F2 đạt giá trị lớn nhất là 4 vậy kết quả bài toán là (F1+F2) -1 (phần tử giữa xuất hiện hai lần).

d. Cài đặt

```
Program wavio;
Const
        limit=10000;
Var
        A,f1,f2:array[1..limit] of integer;
        i,j,n,max:integer;
        fi,fo:text;
{-------------------------------------------------------}
Procedure inputdata;
```

```
    Begin
        assign(fi,'wavio.inp');
        reset(fi);
        readln(fi,n);
        for i:=1 to n do read(fi,a[i]);
        close(fi);
    End;
{-----------------------------------------------------}
Procedure outputdata;
    Begin
        assign(fo,'wavio.out');
        rewrite(fo);
        writeln(fo,max-1);
  close(fo);
    End;
{-----------------------------------------------------}
Procedure process_f1;
    Begin
        f1[1]:=1;
        for i:=2 to n do
            Begin
                f1[i]:=1;
                for j:=1 to i-1 do
                    if (a[i]>a[j]) and (f1[i]<f1[j]+1)
                        then f1[i]:=f1[j]+1;
            End;
    End;
{-----------------------------------------------------}
Procedure process_f2;
    Begin
        f2[n]:=1;
        for i:=n-1 downto 1 do
            Begin
                f2[i]:=1;
                for j:=n downto i+1 do
                    if (a[i]<a[j]) and (f2[i]<f2[j]+1)
                        then f2[i]:=f2[j]+1;
            End;
 End;
{-----------------------------------------------------}
```

```
Procedure result;
    Begin
        max:=0;
        for i:=1 to n do if max < (f1[i]+f2[i])
            then max:=f1[i]+f2[i];
    End;
{-------------------------------------------------------}
Begin
    inputdata;
    process_f1;
    process_f2;
    result;
    outputdata;
End.
```

BÀI 3: LỚP BÀI TOÁN GHÉP CẶP

I. Tổng Quan

1. Mô Hình

Người ta muốn cắm một số bó hoa vào các lọ. Có **W** bó hoa và **V** lọ hoa (W≤V), các lọ hoa được đánh dấu từ 1 đến **V** theo thứ tự từ trái qua phải. Mỗi bó hoa được đánh dấu từ 1 đến **W**. Người ta cắm hoa theo quy tắc sau: Nếu bó hoa **i** cắm vào lọ **Vi**, bó hoa **j** cắm vào lọ **Vj**, với **i**<**j** thì **Vi**<**Vj**.

Nếu bó hoa **i** được cắm vào lọ hoa **j** thì ta có điểm thẩm mỹ là **Aij** (1≤i≤W; 1≤j≤V).

Example

Lọ hoa / Hoa	1	2	3	4	5
Bó 1	7	23	-5	-24	16
Bó 2	5	21	-4	10	23
Bó 3	-21	5	-4	-20	20

Chú ý: Mỗi bó hoa chỉ cắm vào một lọ và mỗi lọ chỉ được cắm tối đa một bó hoa.

Yêu cầu: Bạn hãy giúp mọi người cắm hết hoa vào lọ để tổng điểm thẩm mỹ là cao nhất. Giới hạn: 1 ≤ W, V ≤ 1000; –500 ≤ Aij ≤ 500.

Dữ liệu vào từ tệp văn bản **CAMHOA.INP**:

- Dòng đầu ghi hai số **W**, **V**.
- **W** dòng tiếp theo, mỗi dòng ghi **V** số nguyên, như vậy số **Aij** ghi tại vị trí **j** dòng **i**+1.

Dữ liệu ra ghi vào tệp văn bản **CAMHOA.OUT**: Một số nguyên duy nhất là tổng điểm thẩm mỹ cao nhất.

Example

CAMHOA.INP	**CAMHOA.OUT**
3 5 7 23 -5 -24 16 5 21 -4 10 23 -2 5 -4 -20 20	53

Như ví dụ trên ta có thể chọn bó 1 cắm vào lọ 2, bó 2 cắm vào lọ 4, bó 3 cắm vào lọ 5.

2. Hướng dẫn giải

Giá trị thẩm mỹ phụ thuộc vào số lượng các bó hoa và số lượng lọ hoa đang được xét nên cách cắm (hàm mục tiêu) sẽ phụ thuộc vào hai đại lượng biến thiên đó. Ta sẽ dùng mảng 2 chiều để lưu trữ bảng phương án.

Gọi F[i,j] là tổng giá trị thẩm mỹ lớn nhất khi xét đến bó hoa thứ i và lọ thứ j. Chúng ta có các lựa chọn cách cắm như sau:

Nếu cắm bó hoa i vào lọ j thì: Tổng giá trị thẩm mỹ là **F[i,j] = F[i-1,j-1]+a[i,j]** (Bằng tổng giá trị trước khi cắm cộng với giá trị thẩm mỹ khi cắm bó hoa i vào lọ j).

Không cắm bó hoa i vào lọ j (có thể cắm vào lọ trước j), giá trị của cách cắm này là như cũ: **F[i,j] = F[i,j-1]**.

Chú ý: Nếu i=j (số bó hoa bằng số lọ hoa) thì chỉ có một cách cắm vậy:

F[i,j] =a[1,1]+a[2,2]+…+a[i,i]

Nếu i>j thì không có cách cắm hợp lý: **F[i,j] = 0**.

Vậy công thức truy hồi của chúng ta như sau:

- Nếu i=0 or j=0 thì: F[i,j]=0 (hiển nhiên).
- Nếu i=j thì: **F[i,j] = a[1,1]+a[2,2]+…+a[i,i].**
- Nếu i > j thì: F[i,j] = 0.
- Nếu i < j thì: **F[i,j] = max(F[i-1,j-1]+a[i,j], F[i,j-1])** (chọn lựa nên hay không nên cắm bó i vào lọ j).

Kết quả bài toán: **F[W,V].**

3. Bảng phương án:

Ta xây dựng bảng phương án dựa trên công thức truy hồi trên. Để kiểm tra kết quả có chính xác hay không (nếu không chính xác chúng ta xây dựng lại hàm mục tiêu). Thông qua cách xây dựng hàm mục tiêu và bảng phương án chúng ta sẽ định hướng việc truy vết.

W,V	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	7	23	23	23	23
2	0	0	28	28	33	46
3	0	0	0	24	24	53

4. Truy vết

Nếu yêu cầu hiển thị cách cắm thì chúng ta có thể truy vết theo cách sau:

Xuất phát từ ô F[W,V]:

- B1: Nếu F[i,j]=F[i,j-1] thì truy về ô F[i,j-1] cứ tiếp tục như vậy cho đến khi nào F[i,j] ≠ F[i,j-1] thì thôi. Kết thúc quá trình ghi nhận kết quả tại vị i,j.
- B2: Nếu F[i,j] ≠ F[i,j-1] thì ghi nhận kết quả tại vị trí i,j sau đó truy về ô F[i-1,j-1] rồi quay về B1.

5. Cài Đặt

```
Program camhoa;
const
        limit=1000;
Var
        A,B:array[1..limit,1..limit] of integer;
        F:array[0..limit,0..limit] of longint;
        i,j,W,V,sum:integer;
        fi,fo:text;
{-----------------------------------------------------------------------------------------}
Procedure inputdata;
        Begin
                assign(fi,’ camhoa.inp’);
                reset(fi);
                readln(fi,W,V);
                for i:=1 to F do
                  begin
                              For j:=1 to V do read(fi,a[i,j]);
                        readln(fi);
                  end;
           close(fi);
        End;
{-----------------------------------------------------------------------------------------}
Procedure outputdata;
        Begin
                assign(fo,'camhoa.out');
                rewrite(fo);
                writeln(fo,F[W,V]);
                close(fo);
        End;
{-----------------------------------------------------------------------------------------}
Function max(a,b:longint):longint;
        Begin
                if a>b then max:=a else max:=b;
        End;
{-----------------------------------------------------------------------------------------}
```

```
Procedure sums;{trường tính các giá trị tại vị trí i=j}
        Begin
                b[1,1]:=a[1,1];
                for i:=2 to W do b[i,i]:=b[i-1,i-1]+a[i,i];
        End;
{--------------------------------------------------------------------------------------------}
Procedure process;
        Begin
                for i:=1 to W do
                        for j:=i to V do
                                if i=j then F[i,j]:=b[i,i]
                                 else F[i,j]:=max(F[i-1,j-1]+a[i,j],F[i,j-1]);
        End;
{--------------------------------------------------------------------------------------------}
Begin
        inputdata;
        sums;
        process;
        outputdata;
End.
```

II. Áp Dụng

Bài toán : Bố trí phòng học

a. Phát biểu bài toán

Có n phòng học chuyên đề và k nhóm học được đánh số thứ tự từ nhỏ đến lớn. Cần xếp k nhóm trên vào n phòng học sao cho nhóm có số hiệu nhỏ được xếp vào phòng có số hiệu nhỏ, nhóm có số hiệu lớn phải được xếp vào phòng có số hiệu lớn.Với mỗi phòng có chứ học sinh, các ghế thừa phải được chuyển ra hết, nếu thiếu ghế thì lấy vào cho đủ ghế. Biết phòng i có a(i) ghế, nhóm j có b(j) học sinh. Hãy chọn 1 phương án bố trí sao cho tổng số lần chuyển ghế ra và vào là ít nhất.

Inputdata: Từ file văn bản **Botriphonghoc.inp**

- Dòng 1 : 2 Số nguyên dương n, k. ($1\leq n \leq k \leq 10000$)
- Dòng 2 : n số nguyên chỉ số ghế trong các phòng.
- Dòng 3 : k số nguyên chỉ số học sinh của các nhóm.

Outputdata: Từ file văn bản **Botriphonghoc.out** chứa 1 số duy nhất là kết quả bài toán.

Example

Botriphonghoc.inp	Botriphonghoc.out

5 3 25 30 35 40 45 30 25 40	**10**

b. Hướng dẫn giải

Coi các "nhóm học" là các "bó hoa", các "lớp học" là các "lọ hoa". Vậy cần cắm những "bó hoa" vào các "lọ hoa" sao cho dạt giá trị thẩm mỹ là **thấp nhất**. Giá trị thẩm mỹ ở đây chính là độ chênh lệch số ghế trong phòng học i so với nhóm j (đặt c[i,j]=│a[i]-b[j]│).

c. Cài đặt

```
Program Botriphonghoc;
Const
        limit=1000;
Var
        A,B:Array[1..limit] of integer;
        C,D: Array[1..limit,1..limit] of integer;
        F:Array[0..limit,0..limit] of longint;
        i,j,n,k:integer;
        fi,fo:text;
{-----------------------------------------------------------------------------------------}
Procedure inputdata;
        Begin
                assign(fi,'Botriphonghoc.inp');
                reset(fi);
                readln(fi,n,k);
                for i:=1 to n do read(fi,a[i]); readln(fi);
                for i:=1 to k do read(fi,b[i]);
                close(fi);
                for i:=1 to k do
                        for j:=1 to n do c[i,j]:=abs(a[j]-b[i]);
        End;
{-----------------------------------------------------------------------------------------}
Procedure outputdata;
        Begin
                assign(fo,'botriphonghoc.out');
                rewrite(fo);
                writeln(fo,F[k,n]);
            close(fo);
        End;
{-----------------------------------------------------------------------------------------}
```

```
Procedure sums;{trường tính các giá trị tại vị trí i=j}
    Begin
        d[1,1]:=c[1,1];
        for i:=2 to n do d[i,i]:=d[i-1,i-1] + c[i,i];
    End;
{-----------------------------------------------------------------------------------------}
Function min(x,y:longint):longint;
    Begin
        if x<y then min:=x else min:=y;
    End;
{-----------------------------------------------------------------------------------------}
Procedure process;
    Begin
        for i:=1 to k do
            for j:=i to n do
                if i=j then F[i,j]:=d[i,i]
                else F[i,j]:=min(F[i-1,j-1]+c[i,j],F[i,j-1]);
    End;
{-----------------------------------------------------------------------------------------}
Begin
    inputdata;
    sums;
    process;
    outputdata;
End.
```

BÀI 4: LỚP BÀI TOÁN DI CHUYỂN

I. Tổng Quan

a. Mô Hình

Một người may mắn gặp một ma trận kim cương gồm MxN ô. Giá trị A[i,j] (A[i,j] là một số nguyên dương) là lượng kim cương có ở ô ở dòng i cột j. Người này chỉ được xuất phát tử một ô ở mép bên trái của ma trận và di chuyển sáng mép bên phải. Từ ô (i,j) người này chỉ có thể di chuyển sang 1 trong 3 ô (i-1,j+1), (i,j+1) hoặc (i+1,j+1). Di chuyển qua ô nào thì người đó được phép mang theo lượng kim cương ở ô đó. Em hãy giúp người này di chuyển theo đường đi nào để có thể nhận được nhiều kim cương nhất.

InputData: file văn bản **vanmay.inp**

- Dòng thứ nhất gồm hai số nguyên M, N ($0 \le M, N \le 5000$) cách nhau 1 khoảng trắng
- M dòng tiếp theo mỗi dòng ghi ghi một hàng các số nguyên của ma trận (các số nguyên có giới hạn từ 10^9).

OutputData: file văn bản **vanmay.out**: Ghi tổng lượng kim cương nhiều nhất có thể có được.

Example

VanMay.inp	VanMay.out
2 3 1 2 5 3 4 6	12

b. Hướng dẫn giải

Gọi F[i,j] là giá trị lớn nhất có được khi di chuyển đến ô (i,j). Có 3 ô có thể di chuyên đến ô (i,j) là ô (i,j-1), (i-1,j-1) và (i+1,j-1).

Vậy chúng ta có công thức truy hồi như sau:

- F[1,j]=A[1,j]
- F[i,j]=max(F(i,j-1),F(i-1,j-1),F(i+1,j-1))+A[i,j] với i>1.

c. Bảng phương án

i/j	*0*	*1*	*2*	*3*
0	0	0	0	0
1	0	1	5	13
2	0	3	8	14
3	0	4	6	13

d. Truy vết.

- Xuất phát từ ô kết quả (ô đạt giá trị lớn nhất ở bìa phải ma trận).
- Tại ô(i,j) đi qua trái chọn ô lớn nhất trong 3 ô

e. Cài đặt

```
Program VanMay;
Const
        Limit = 5000;
Var
        fi,fo:text;
        A:Array[1..limit,1..limit] of longint;
        F:Array[0..limit,0..limit] of longint;
        i,j,n,m: integer;
        emax:int64;
{--------------------------------------------------------------------}
Procedure inputdata;
        Begin
                assign(fi,'vanmay.inp');
                reset(fi);
                readln(fi,n,m);
                for i:=1 to n do
                        begin
                                for j:=1 to m do read(fi,a[i,j]);
                                readln(fi);
                        end;
                close(fi);
        end;
{--------------------------------------------------------------------}
Procedure outputdata;
        Begin
                assign(fo,'vanmay.out');
                rewrite(fo);
                writeln(fo,emax);
            close(fo);
        End;
{--------------------------------------------------------------------}
Function max(a,b:int64): int64;
        Begin
                if a>b then max:=a
                else max:=b;
```

```
        End;
{----------------------------------------------------------------------}
Procedure process;
        Begin
           for j:=1 to m do {tính giá trị của cột trước vì cần sử dụng kết quả của cột}
                    for i:=1 to n do
                            f[i,j]:=max(F[i,j-1],max(F[i-1,j-1],F[i+1,j-1]))+a[i,j];
                emax:=f[n,1];
                for i:=1 to n do
                    if f[i,m]>emax then
                            emax:=f[i,m];
        End;
{----------------------------------------------------------------------}
Begin
        inputdata;
        process;
        outputdata;
End.
```

II. Áp dụng

Bài toán 1: Tam giác số (IOI 1994).

a. Mô hình

				7				
			3		**5**			
		8		**1**		**0**		
	2		***7***		**4**		**4**	
4		***5***		**2**		**6**		**5**

Hình trên biểu diễn tam giác số: Hãy viết chương trình tính tổng lớn nhất các số trên còn đường bắt đầu từ đỉnh và kết thúc ở đáy.

- Mỗi bước có thể đi chéo xuống phía trái hoặc đi chéo xuống phía phải.
- Số lượng hàng trong tam giác lớn hơn 1 nhưng ≤ 3333
- Các số trong tam giác đều là số nguyên từ 0 đến 1 tỉ (10^9).

INPUT DATA: File văn bản **Tamgiacso.inp**:

- Dòng 1: Ghi số lượng dòng của tam giác
- N dòng tiếp theo: Dòng i lưu trữ các số của dòng thứ i trong tam giác.

OUTPUT DATA: File văn bản **tamgiacso.out**: Ghi tổng lớn nhất

Example

Tamgiacso.inp	Tamgiacso.out
5 7 3 5 8 1 0 2 7 4 4 4 5 2 6 5	30

b. Hướng dẫn giải

Gọi F[i,j] là tổng lớn nhất đi từ đỉnh tam giác tới ô (i,j).

Trường hợp đơn giản nhất i=1 và j=1 thì F[i,j] = a[i,j] (A là mảng dữ liệu ban đầu của tam giác).

- Nếu j=1(vị trí bìa trái của tam giác): F[i,j]:=F[i-1,j] + a[i,j].
- Nếu j=i (vị trí bìa phải của tam giác): F[i,j]:=F[i-1,j-1] + a[i,j].

Với các vị trí khác: F[i,j]:=max(F[i-1,j], F[i-1,j-1]) + a[i,j].

Vậy giá trị lớn của đường đi sẽ là kết quả lớn nhất của dòng đáy của tam giác vậy

Max = max(F[n,j]) (j:=1..n).

c. Bảng phương án

i/j	***0***	***1***	***2***	***3***	***4***	***5***
0	0	0	0	0	0	0
1	0	***7***	0	0	0	0
2	0	10	***12***	0	0	0
3	0	***18***	13	12	0	0
4	0	20	***25***	17	16	0
5	0	24	***30***	27	23	21

d. Truy vết

Xuất phát từ ô kết quả (ô có giá trị lớn nhất ở đáy tam giác)

Tại ô (i,j) ta đi ngược lên chọn ô lớn trong 2 ô kề trên nó (i-1,j-1) và (i-1,j).

e. Cài đặt

```
Program Tamgiacso;
Const
        Limit = 5000;
Var
        fi,fo:text;
        A:Array[1..limit,1..limit] of longint;
        F:Array[0..limit,0..limit] of longint;
        i,j,n: integer;
        emax:int64;
{-------------------------------------------------------------------}
Procedure inputdata;
        Begin
                assign(fi,'tamgiacso.inp');
                reset(fi);
                readln(fi,n);
                for i:=1 to n do
                        begin
                                for j:=1 to i do read(fi,a[i,j]);
                                readln(fi);
                        end;
                close(fi);
        end;
{-------------------------------------------------------------------}
Procedure outputdata;
        Begin
                assign(fo,'tamgiacso.out');
                rewrite(fo);
                writeln(fo,emax);
            close(fo);
        End;
{-------------------------------------------------------------------}
Function max(a,b:int64): int64;
        Begin
                if a>b then max:=a
                else max:=b;
        End;
```

```
{-------------------------------------------------------------------}
Procedure process;
	Begin
		for i:=1 to n do
			for j:=1 to i do
				f[i,j]:=max(f[i-1,j],f[i-1,j-1])+a[i,j];
		emax:=f[n,1];
		for j:=1 to n do
			if f[n,j]>emax then
				emax:=f[n,j];
	End;
{-------------------------------------------------------------------}
Begin
	inputdata;
	process;
	outputdata;
End.
```

BÀI 5: DẠNG BÀI TOÁN BIẾN ĐỔI XÂU

I. Tổng quan

1. Mô Hình: Cho hay xâu X và Y:

- X gọi là xâu nguồn, X có n ký tự: $X = <x_1, x_2,\ldots,x_n>$.
- Y gọi là xâu đích, Y có m ký tự: $Y = <y_1, y_2,\ldots,y_m>$.

Có 3 phép biến đổi xâu như sau:

- Chèn một ký tự vào sau ký tự thứ i.
- Thay thế ký tự ở vị trí thứ i bằng ký tự khác.
- Xóa ký tự ở vị trí thứ i.

Yêu cầu: Hãy tìm số ít nhất các phép biến đổi để biến xâu X thành xâu Y.

2. Hướng dẫn

Ta thấy số phép biến đổi phụ thuộc vào vị trí i đang xét của xâu X và vị trí j đang xét của xâu Y. Do hàm mục tiêu của chúng ta sẽ phụ thuộc vào hay tham số biến thiên là i và j. Vì thế để cài đặt cho bảng phương án ta cần sử dụng mảng hai chiều.

Gọi F[i,j] là số phép biến đổi ít nhất để biến đổi xâu X(i) ($X = <x_1, x_2,\ldots,x_i>$: i ký tự đầu tiên của xâu X) thành xâu Y(j) ($Y = <y_1, y_2,\ldots,y_j>$: j ký tự đầu tiên của xâu Y). Vậy ta dễ thấy F[0,j] = j và F[i,0] = i.

Khi chúng ta xét hai vị trí i và j thì sẽ có hay trường hợp xẩy ra như sau:

Trường hợp 1: Nếu $x_i = y_j$ thì bài toán lúc này của chúng ta trở thành bài toán biến đổi xâu X(i-1) thành xâu Y(j-1). Do đó F[i,j] = F[i-1,j-1].

Trường hợp 2: Nếu $x_i \neq y_j$ thì lúc này chúng ta sẽ có 3 cách biến đổi như sau:

- **Cách 1**: Xóa ký tự x_i thì lúc này bài toán trở thành bài toán biến đổi xâu X(i-1) thành xâu Y(j). Do đó F[i,j] = F[i-1,j] + 1 (cộng 1 là do chúng ta đã dùng 1 phép xóa).
- **Cách 2**: Thay thế ký tự $\mathbf{x_i}$ bằng ký tự $\mathbf{y_j}$ ($X= <x_1, x_2,\ldots, \mathbf{y_j}>$, $Y = <y_1, y_2,\ldots,y_j>$). Lúc này bài toán trở thành bài toán biến đổi xâu X(i-1) thành xâu Y(j-1). Do đó F[i,j]=F[i-1,j-1]+1 (cộng 1 do chúng ta đã dùng một phép thay thế).
- **Cách 3**: Chèn ký tự y_j vào vị trí x_i ($X= <x_1, x_2,\ldots, x_i, \mathbf{y_j}>$, $Y = <y_1, y_2,\ldots,y_j>$). Thì lúc này bào toán trở thành bài toán biến đổi xâu X(i) thành xâu Y(j-1). Do đó F[i,j]=F[i,j-1] +1.

Vậy chúng ta có công thức QHD tổng quát như sau:

- F[0,j] = j.
- F[i,j] = i.
- F[i,j] = F[i-1,j-1] nếu X[i] = Y[j].
- F[i,j] = min(F[i-1,j], F[i-1,j-1], F[i,j-1]) +1 nếu X[i] ≠ Y[j].

3. Cải tiến

Đối với lớp bài toán dạng này nếu không yêu cầu truy vết thì chúng ta có thết kiệm biến hơn nếu dùng hai mảng một chiều để tính lẫn nhau (vì vào mỗi thời điểm chúng ta chỉ cần sử dụng hàng đang tính và hàng trước nó). Còn nếu yêu cầu truy vết thì 2 mảng 1 chiều không lưu trữ đủ thông tin để chúng ta rà lại các bước trong khoảng O(n+m) thời gian.

II. Áp dụng

Bài toán 1: Tìm xâu con chung dài nhất.

a. Phát biểu bài toán

Xâu ký tự A được gọi là xâu con của xâu ký tự B nếu ta có thể xoá đi một số ký tự trong xâu B để được xâu A.

Cho biết hai xâu ký tự X (X = <x_1, x_2,…,x_n>) và Y (Y = <y_1, y_2,…,y_m>), hãy tìm xâu ký tự Z có độ dài lớn nhất và là con của cả X và Y.

InputData: File văn bản **LCS.inp** gồm:

Dòng 1: chứa xâu X

Dòng 2: chứa xâu Y

OutputData: File văn bản **LCS.out:** Chỉ gồm một dòng ghi độ dài xâu Z tìm được

Example

LCS.inp	LCS.Out
ALGORITHM LOGARITHM	7

b. Hướng dẫn giải

Ta thấy độ dài dãy con chung dài nhất phụ thuộc vào vị trí i đang xét của xâu X và vị trí j đang xét của xâu Y. Do đó hàm mục tiêu của chúng ta sẽ phụ thuộc vào hai tham số biến thiên là i và j. Vì thế để cài đặt cho bảng phương án ta cần sử dụng mảng hai chiều.

Giả sử F[i,j] là độ dài độ dài lớn nhất của dãy con chung của hay dãy:

+ $x_1,x_2,\ldots,x_i$

+ $y_1,y_2,\ldots,y_j$

Ta đi tìm công thức truy hồi để tính F[i,j]:

+ Nếu i=0 hoặc j=0 thì F[i,j] = 0;

+ Nếu i>0 và j>0 và $x_i = y_j$ thì F[i,j] = 1 + F[i-1,j-1];

+ Nếu i>0 và j>0 và $x_i \neq y_j$ thì F[i,j] = max(F[i-1,j], F[i,j-1]);

Khi đó F[n,m] chính là độ dài xâu con chung dài nhất của hay xâu X và Y.

c. Bảng phương án và truy vết

		A	L	G	O	R	I	T	H	M
	0	0	0	0	0	0	0	0	0	0
L	0	0	1	1	1	1	1	1	1	1
O	0	0	1	1	2	2	2	2	2	2
G	0	0	1	2	2	2	2	2	2	2
A	0	1	1	2	2	2	2	2	2	2
R	0	1	1	2	2	3	3	3	3	3
I	0	1	1	2	2	3	4	4	4	4
T	0	1	1	2	2	3	4	5	5	5
H	0	1	1	2	2	3	4	5	6	6
M	0	1	1	2	2	3	4	5	6	7

Bảng b mà thủ tục LCS – LENGTH trả về có thể được dùng để nhanh chóng kiến tạo một LCS của X = <x_1, x_2,…,x_m> và Y = <y_1, y_2,…,y_n>. Ta đơn giản bắt đầu từ b[m,n] và rà qua bảng theo các mũi tên. Mỗi khi ta gặp một “□” *trong việc nhập b[i,j], nó hàm ý rằng x_i = y_j là một thành phần của LCS. Phương pháp này in ra một đề xuất đảo ngược. Thủ tục đệ quy sau đây in ra một LCS đúng đắn.*

```
PRINT – LCS (b,X,i, j)

if i =0 or j=0
        Then reture
if b[i,j] =  “□”
        Then Print – LCS (b,X,i-1,j-1);
        Print xi
else if b[i,j]  =  “↑”
        Then Print – LCS (b,X,i-1,j);
else Print – LCS (b,X,i,j-1);
```

Thủ tục PRINT – LCS (b,X,i, j) mất một thời gian O(m+n), bởi ít nhất một trong số biến i, j được giảm lượng trong mỗi giai đoạn đệ quy.

d. Cài đặt

```
Program LCS;
Const
        limit = 250;
Var
        fi,fo : text;
```

```
        s1,s2 : string;
        m,n,i,j : byte;
        F : Array[0..limit,0..limit] of byte;
{---------------------------------------------}
Function max(a,b:byte):byte;
        Begin
                if a>b then max:=a
                else max:=b;
        End;
{---------------------------------------------}
Procedure inputdata;
        Begin
                assign(fi,'lcs.inp');
                reset(fi);
                readln(fi,s1);
                read(fi,s2);
                close(fi);
        End;
{---------------------------------------------}
Procedure outputdata;
        Begin
                assign(fo,'lcs.out');
                rewrite(fo);
                write(fo,F[n,m]);
                close(fo);
        End;
{---------------------------------------------}
Procedure process;
        Begin
                n:=length(s1);
                m:=length(s2);
                for i:= 1 to n do
                        for j:=1 to m do
                                if s1[i]=s2[j] then
                                        F[i,j]:= 1+ F[i-1,j-1]
                                else
                                        F[i,j]:=max(F[i-1,j],F[i,j-1]);
        End;
{---------------------------------------------}
Begin
```

inputdata;
process;
outputdata;
End.

Chú ý: Đối với những bài toán tối ưu dạng này thường rất ít khi yêu cầu chúng ta truy vết, bởi vì có nhiều con đường dẫn đến kết quả tối ưu cho nên rất khó để kiểm tra.

Bài toán 2: Bài Toán Bắc cầu

a. Mô hình

Sông Mê Công là một trong những con sông lớn nhất trên thế giới, bắt nguồn từ Trung Quốc, chảy qua Lào, Myanma, Thái Lan, Campuchia và đổ ra Biển Đông ở Việt Nam. Và nó cũng tạo ra một đường biên giới giao thương lý tưởng cho hai nước Lào và Thái Lan. Ở hai bên bờ sông Mê Công, Lào nằm ở bờ Đông và có **N** tỉnh được đánh số từ 1 đến **N**, Thái Lan nằm ở bờ Tây có **M** tỉnh được đánh số từ 1 đến **M** (theo vị trí từ Bắc xuống Nam). Mỗi tỉnh của nước này thường có quan hệ kết nghĩa với một số tỉnh của nước kia. Để tăng cường tình hữu nghị, hai nước muốn xây dựng các cây cầu bắc qua sông, mỗi cây cầu sẽ là nhịp cầu nối 2 tỉnh kết nghĩa. Để xây dựng được các cây cầu thì phải thỏa một số điều kiện sau:

- Các cây cầu không được cắt nhau;
- Mỗi tỉnh là đầu cầu nhiều nhất cho một cây cầu.

Yêu cầu: Bạn hãy tính toán giúp hai nước Lào và Thái Lan xem có thể xây dựng tối đa bao nhiêu cây cầu.

Dữ liệu vào từ tệp văn bản **BACCAU.INP**:

- Dòng đầu tiên: chứa ba số nguyên dương **N**, **M**, **K** (0 < N, M, K < 10000);
- **K** dòng sau: mỗi dòng chứa hai số **i**, **j** thể hiện sự kết nghĩa của tỉnh **i** (bờ Đông) với tỉnh **j** (bờ Tây).

Các số trên một dòng được phân cách bởi khoảng trắng.

Dữ liệu ra ghi vào tệp văn bản **BACCAU.OUT**: Một số nguyên duy nhất là số cây cầu tối đa có thể xây dựng.

Ví dụ:

BACCAU.INP	BACCAU.OUT
5 5 10 3 4 4 2 4 4 5 1	4

5 2 5 3 3 2 5 5 1 3 2 1	

Giải thích ví dụ: có thể chọn xây các cầu 2–1, 3–2, 4–4, 5–5.

b. Hướng dẫn giải

Gọi các tỉnh của Lào lần lượt là a1, a2,…, an, các tỉnh của Thái Lan là b1,b2,…,bm. Nếu tỉnh ai và bj kết nghĩa với nhau thì coi ai "bằng" bj. Để các cây cầu không cắt nhau, trong khi ta đã chọn cặp tỉnh (ai,bj) để xây cầu thì cặp tiếp theo phải là cặp (au,bv) sao cho u>i và v>j. Như vậy các cặp tỉnh được chọn xây cầu có thể coi là một dãy con chung dài nhất của hay dãy a và b.

Bài toán của chúng ta trở thành bài toán tìm dãy con chung dài nhất, ở đây hai phần tử "bằng" nhau nếu chúng có quan hệ kết nghĩa.

Để đánh dấu những tỉnh nào có kết nghĩa với nhau chúng ta sử dụng một mảng hai chiều kiểu boolean để đánh dấu. Ví dụ nếu tỉnh thứ 2 của Lào kết nghĩa với tính thứ 3 của Thái Lan thì A[2,3] = True (quan hệ "bằng" trong xâu con chung dài nhất).

c. Cài đặt

```
Program BacCau;
const
        limit = 10000;
Var
        fi,fo :text;
        F: Array[0..limit,0..limit] of longint;
        A: Array[1..limit,1..limit] of boolean;
        i,j,x,y,n,m,k: longint;
{-------------------------------------------------------------------------}
Procedure inputdata;
        Begin
                assign(fi,'baccau.inp');
                reset(fi);
            readln(fi,n,m,k);
                for i:=1 to k do
                        Begin
                                readln(fi,x,y);
```

```
                        a[x,y]:=true;
                    end;
                close(fi);
        End;
{--------------------------------------------------------------------------}
Procedure outputdata;
        Begin
                assign(fo,'baccau.out');
                rewrite(fo);
                write(fo,F[n,m]);
                close(fo);
        End;
{--------------------------------------------------------------------------}
Function max(a,b:integer):integer;
        Begin
                if a>b then max:=a
                else max:=b;
        End;
{--------------------------------------------------------------------------}
Procedure process;
        Begin
                for i:=1 to n do
                    for j:=1 to m do
                        if a[i,j]=true then
                            F[i,j]:=1+F[i-1,j-1]
                        else
                            F[i,j]:=max(F[i-1,j],F[i,j-1]);
        End;
{--------------------------------------------------------------------------}
Begin
        inputdata;
        process;
        outputdata;
End.
```

Bài toán 3: Bài toán phá cầu

a. Mô hình

Để giải quyết tai nạn giao thông trên đường thủy của đất nước Olympic, ủy ban quốc gia nước này thấy cần phải phá hủy 1 số cây cầu cắt nhau trên dòng sông A. Có n (n <10000) cây cầu được xây dựng dọc theo hai bờ sông. Mỗi cây cầu được biểu thị 2 số ai, bi (0 < ai, bi < 10000, i= 1..n), trong đó:

- ai, bi tương ứng là 2 điểm nằm trên hai bờ sông.
- Hai cây cầu k va l được gọi là cắt nhau khi ak ≤ al và bk ≥ bl hoặc ak ≥ al và bk ≤ bl (0<k, l<n+1).

Yêu cầu

- Không còn hai cây cầu nào cắt nhau trên dòng sông.
- Số cây cầu cần phải phá bỏ là ít nhất.

InputData: File văn bản tên là **PhaCau.inp** có cấu trúc như sau:

- Dòng đầu là số nguyên dương n.
- n dòng tiếp theo mỗi dòng chứa 2 số ai, bi là tọa độ tương ứng của n cây cầu.

OutputData: File văn bản tên là **PhaCau.out** chứa 1 số duy nhất là số lượng cây cầu cần phá (ít nhất).

Ví dụ:

Phacau.inp	phacau.inp
10 3 4 4 2 4 4 5 1 5 2 5 3 3 2 5 5 1 3 2 1	6

b. Hướng dẫn giải

Đối với một số bài toán việc đi tìm B có thể gặp nhiều khó khăn, nhưng việc tìm A lại dễ dàng hơn. Mà từ A có thể suy ra B. Vậy thì thay vì đi tìm B chúng ta trở về bài toán tìm A.

Đối với bài toán này thay vì đi tìm số cây cầu phải phá, chúng ta sẽ tìm số cây cầu nhiều nhất không cắt nhau. Vậy dễ dàng thấy B = N – A.

Vậy bài toán trở thành bài toán tìm số cây cầu lớn nhất có thể bắc sao cho không cây cầu nào cắt nhau.

c. Cài đặt

```
Program PhaCau;
const
        limit = 10000;
Var
        fi,fo :text;
        F: Array[0..limit,0..limit] of longint;
        A: Array[1..limit,1..limit] of boolean;
        i,j,x,y,n: longint;
{------------------------------------------------------------------------}
Procedure inputdata;
        Begin
                assign(fi,'phacau.inp');
                reset(fi);
            readln(fi,n);
                for i:=1 to n do
                        Begin
                                readln(fi,x,y);
                                a[x,y]:=true;
                        end;
                close(fi);
      End;
{------------------------------------------------------------------------}
Procedure outputdata;
        Begin
                assign(fo,'phacau.out');
                rewrite(fo);
                write(fo,n-F[n,n]);
                close(fo);
        End;
{------------------------------------------------------------------------}
Function max(a,b:integer):integer;
        Begin
                if a>b then max:=a
                else max:=b;
        End;
{------------------------------------------------------------------------}
Procedure Process;
        Begin
                for i:=1 to n do
```

```
                    for j:=1 to n do
                              if a[i,j]=true then
                                        F[i,j]:=1+F[i-1,j-1]
                              else
                                        F[i,j]:=max(F[i-1,j],l[i,j-1]);
          End;
{---------------------------------------------------------------------------}
Begin
          inputdata;
          Process;
          outputdata;
End.
```

Bài toán 4: Chuỗi con đối xứng

a. Mô hình

Một chuỗi được gọi là đối xứng nếu nó không có ít hơn một ký tự và nếu ta đọc từ trái san g phải hay từ phải sang trái đều giống nhau.

Example : ‘A’ ; ‘TET’ ; ‘CAOOAC’ là chuỗi đối xứng

‘BHABHCD’ là chuỗi không đối xứng

Viết chương trình nhập vào chuỗi ký tự cho trước S, có chiều dài n (1<=n<=20000) và cho biết chiều dài chuỗi con đối xứng dài nhất. Chuỗi con của S là chuỗi gồm 1 số ký tự có thứ tự trong S có độ dài nhỏ hơn hoặc bằng n.

<u>**Dữ liệu vào**</u> được cho trong tập tin văn bản **CCDX.INP** gồm 2 dòng :

- Dòng đầu ghi giá trị n
- Dòng sau gồm n ký tự liên tiếp gồm các chữ cái in hoa (A → Z)

<u>**Dữ liệu ra**</u> : Ghi vào tập tin văn bản **CCDX.OUT** gồm 1 số duy nhất là độ dài của chuỗi con đối xứng dài nhất.

Example 1 :

CCDX.INP	CCDX.OUT
18 IKACOBEGIGEBOCAHTM	13

Example 2 :

CCDX.INP	CCDX.OUT

19 IKACOBEGIGEMHBEGIGE	11

b. Hướng dẫn giải

Chúng ta tìm chuỗi đảo ngược của chuỗi S là chuỗi S1. Dễ dàng thấy độ dài chuỗi con đối xứng dài nhất chính là độ dài chuỗi con chung dài nhất của S và S1. Vậy bài toán trở thành bài toán tìm chuỗi con chung dài nhất của S và S1.

c. Cài đặt

```
Program CCDX;
Const
        limit = 250;
Var
        fi,fo : text;
        s1,s : string;
        n,i,j : byte;
        F : Array[0..limit,0..limit] of byte;
{---------------------------------------------}
Procedure inputfile;
        Begin
                assign(fi,'ccdx.inp');
                reset(fi);
                readln(fi,n);
                read(fi,s);
                close(fi);
        End;
{---------------------------------------------}
Procedure outputfile;
        Begin
                assign(fo,'ccdx.out');
                rewrite(fo);
                write(fo,F[n,n]);
                close(fo);
        End;
{---------------------------------------------}
Function max(a,b:byte):byte;
        Begin
                if a>b then max:=a
                else max:=b;
        End;
```

```
{---------------------------------------------}
Procedure chuoidaonguoc;
        Begin
                j:=1;
                for i:=n downto 1 do
                        Begin
                                s1[j]:=s[i];
                                j:=j+1;
                        End;
        End;
Procedure Process;
        Begin
            chuoidaonguoc;
                for i:= 1 to n do
                        for j:=1 to n do
                                if s[i]=s1[j] then
                                        F[i,j]:= 1+ F[i-1,j-1]
                                else
                                        F[i,j]:=max(F[i-1,j],F[i,j-1]);
        End;
{---------------------------------------------}
Begin
        inputfile;
        Process;
        outputfile;
End.
```

Bài toán 5: Palindrom.

a. Mô hình

Một xâu gọi là xâu đối xứng (palindrom) nếu xâu đó đọc từ trái sang phải hay từ phải sang trái đều như nhau.

Yêu cầu : Cho một xâu S, hãy tìm số kí tự ít nhất cần thêm vào S để S trở thành xâu đối xứng.

Inputdata: file văn bản **Palin.inp**: Gồm một dòng duy nhất là xâu S (không phân biệt chữ hoa và chữ thường).

Outputdata: file văn bản **Palin.out**: Gồm một số duy nhất là số kị tự cần thêm vào S hoặc -1 nếu không cần thêm một kí tự nào.

Example:

Palin.inp	Palin.out
edbabcd	2

Giải thích: 2 kí tự cần thêm vào là e và c.

b. Hướng dẫn giải:

Gọi F(i,j) là số kí tự ít nhất cần thêm vào xâu con S[i..j] của S để xâu đó trở thành đối xứng.

Đáp số của bài toán sẽ là F(1,n) với n là số kí tự của S. Ta có công thức sau để tính F(i,j):

- F(i,i)=0.
- F(i,j)=F(i+1,j–1) nếu S[i]=S[j]
- F(i,j)=min(F(i+1,j), F(i,j–1)) nếu S[i]≠S[j]

Ta có thuật toán đơn giản hơn như sau:

Gọi P là xâu đảo của S và T là xâu con chung dài nhất của S và P. Khi đó các kí tự của S không thuộc T cũng là các kí tự cần thêm vào để S trở thành đối xứng. Đáp số của bài toán sẽ là n–k, với k là độ dài của T.

<u>Ví dụ</u>: S=edbabcd, xâu đảo của S là P=dcbabde. Xâu con chung dài nhất của S và P là T=dbabd. Như vậy cần thêm 2 kí tự là e và c vào để S trở thành xâu đối xứng.

c. Cài đặt

```
program Palin;
Const maxn = 5000;
Var
        n,i,j,t: longint;
        f: array[0..maxn, 0..maxn] of longint;
        s: ansistring;
        fi,fo: text;
{-------------------------------------------------------------------------------}
Procedure inputdata;
      Begin
            assign(fi, 'palin.inp');
            reset(fi);
            readln(fi, s);
            n:= length(s);
            close(fi);
            End;
{-------------------------------------------------------------------------------}
Procedure outputdata;
      Begin
            assign(fo, 'palin.out');
```

```
        rewrite(fo);
        write(fo, f[1, n]);
        close(fo);
    End;
{--------------------------------------------------------------------------------}
Function min(x,y: longint): longint;
    Begin
        if x> y then exit(y)
        else exit(x);
    End;

{--------------------------------------------------------------------------------}
Procedure Process;
    Begin
        for t:= 2 to n do
            for i:= 1 to n - t + 1 do
                Begin
                    j:= i + t - 1;
                    if s[i] = s[j] then
                        f[i,j]:= f[i+1, j-1];
                    if s[i] <> s[j] then
                        f[i,j]:= min(f[i + 1, j], f[i, j - 1]) + 1;
               End;
     End;
{--------------------------------------------------------------------------------}
Begin
    inputdata;
    Process;
    outputdata;
End.
```

BÀI 6: LỚP BÀI TOÁN DÃY CON CÓ TỔNG BẰNG S

I. TỔNG QUAN

1. Mô hình

Cho dãy n số nguyên: a_1, a_2, …,a_n. Hãy chỉ ra một dãy con của dãy đã cho có tổng bằng S. Giới hạn $a_i \le 2^{10}$, $n \le 1000$

Inputdata: file văn bản **sums.inp**

- Dòng 1: Chứ 2 số n và S cách nhau ít nhất 1 khoảng trắng
- Dòng 2: Gồm n số nguyên, mỗi số cách nhau ít nhất 1 khoảng trắng

Outputdata: file văn bản **sums.out**:

- Dòng 1: Chứa đáp án "yes" or "no"
- Dòng 2: dãy con

Example:

SumS.inp	SumS.out
3 6 3 2 4	yes 2 4

2. Hướng dẫn giải

Nếu tồn tại một dãy con có tổng bằng S thì ta xét 2 khả năng xẩy ra như sau:

- Nếu phần tử a_n được chọn thì: Nếu tìm được dãy con của n-1(loại bỏ phần từ a_n) số nguyên sao cho có tổng bằng S-a_n thì bài toán ban đầu sẽ sẽ tìm được. Tương tự cho các phần tử ai khác nếu được chọn.
- Nếu phần tử a_n không được chọn thì: Nếu tìm được dãy con của n-1(loại bỏ phần từ a_n) số nguyên sao cho có tổng bằng S thì bài toán ban đầu sẽ sẽ tìm được. Tương tự cho các phần tử ai khác nếu không được chọn.

Vậy chúng ta thấy rằng sẽ có hai đại lượng biến thiên là số lượng phần tử và tổng. Từ đó cho ta thấy rằng hàm mục tiêu của chúng ta sẽ phụ thuộc vào hai đại lượng biến thiên. Do vậy bảng phương án của chúng ta sẽ là bảng hai chiều.

Gọi F[i,T] là trạng thái có thể chọn được (không được chọn) dãy con của dãy từ a_1 đến a_i có tổng bằng T.

- F[i,T)=0 nếu không có dãy con của dãy từ a_1 đến a_i có tổng bằng T.
- F[i,T]=1 nếu có dãy con của dãy từ a_1 đến a_i có tổng bằng T.

Vậy nếu F[n,S]=1 thì có tồn dãy con từ dãy ban đầu có tổng bằng S.

Vậy ta có công thức truy hồi như sau:

- F[i,0]=1 (quy ước).
- F[0,t]=0 (hiển nhiên).
- F[i,t]=1 ⇔ $\left[\begin{array}{c} F[i-1,t]=1 \\ F[i-1,t-a[i]]=1 \end{array}\right.$

3. Bảng phương án

Ta xây dựng bảng phương án dựa trên công thức truy hồi trên. Để kiểm tra kết quả có chính xác hay không (nếu không chính xác chúng ta xây dựng lại hàm mục tiêu). Thông qua cách xây dựng hàm mục tiêu và bảng phương án chúng ta sẽ định hướng việc truy vết.

Example:

SumS.inp	SumS.out
3 6 3 2 4	yes 2 4

Bảng phương án:

N/S	***0***	***1***	***2***	***3***	***4***	***5***	***6***
0	1	0	0	0	0	0	0
1	1	0	0	1	0	0	0
2	1	0	1	1	0	1	0
3	1	0	1	1	1	1	1

Vậy chúng ta chọn được một đãy con là phần hai phần tử 2 4.

4. Truy vết

Nếu F[n,S]=1 thì thông báo “yes”, ngược lại “No”.

Để hiện thị các phần tử trong dãy con ta thực hiện các bước sau:

- Gán i:=n, t:=S;
- Đầu tiên từ F[i,t] nếu F[i-1,t]=1 thì truy đến F[i-1,t] có nghĩa là không chọn phần tử thứ ai
- Nếu F[i-1,t]=0 thì có nghĩa là chọn phần tử ai rồi truy về F[i-1,t-ai].
- Thực hiện đến khi nào i=0.

5. Cài đặt

```
Program DaycontongS;
Const
        limit = 100000;
Var
        F: array[0..100,0..limit] of longint;
        A,b: array[1..100] of longint;
        n,i,k:integer;
        S,t:longint;
        fi,fo:text;
{------------------------------------------------------------------------------------}
Procedure inputdata;
        Begin
```

```
            assign(fi,'sums.inp');
            reset(fi);
            readln(fi,n,s);
            for i:=1 to n do read(fi,a[i]);
            close(fi);
        End;
{-------------------------------------------------------------------------------------}
Procedure outputdata;
        Begin
            assign(fo,'sums.out');
            rewrite(fo);
            if f[n,s]=1 then
                begin
                    writeln(fo,'yes');
                for i:=k downto 1 do write(fo,b[i],' ');
                end
            else write(fo,'no');
            close(fo);
        End;
{-------------------------------------------------------------------------------------}
Procedure process;
        Begin
            for i:=0 to n do F[i,0]:=1;
            for i:=1 to n do
                for t:=1 to s do
                    if (F[i-1,t]=1) then F[i,t]:=1
                else if t-a[i]>=0 then if F[i-1,t-a[i]]=1
                        then F[i,t]:=1;
        End;
{-------------------------------------------------------------------------------------}
Procedure truyvet;
       Begin
           i:=n;  t:=s; k:=0;
           while i>0 do
               begin
                   if F[i-1,t]=1 then i:=i-1
                   else if F[i-1,t-a[i]]=1 then
                       begin
                           k:=k+1;
                           b[k]:=a[i];
                           t:=t-a[i];
                           i:=i-1;
                       end;
               end;
       End;
{-------------------------------------------------------------------------------------}
```

Begin
inputdata;
process;
truyvet;
outputdata;
End.

II. Áp dụng

Bài toán 1: Bài toán chia kẹo

a. Phát biểu bài toán

Cho n gói kẹo, gói thứ i có ai viên. Hãy chia các gói thành 2 phần sao cho độ chênh lệch số kẹo giữa 2 phần là ít nhất.

Inputdata: file văn bản **chiakeo.inp**

- Dòng đầu là n, số gói kẹo ($n \le 10^5$).
- Dòng tiếp theo là n giá trị ai, số viên kẹo của gói thứ i ($ai \le 10^5$).

Outputdata: file văn bản **chiakeo.out**: ghi độ chênh lệch ít nhất giữa hay phần

Example:

CHIAKEO.INP	CHIAKEO.OUT
5 2 4 6 8 10	2

b. Hướng dẫn giải

Gọi T là tổng số kẹo của n gói. Chúng ta cần tìm số S lớn nhất thỏa mãn:

- $S \le T/2$
- Có một dãy con của dãy a có tổng bằng S

Khi đó sẽ có cách chia với chênh lệch 2 phần là T-2*S là nhỏ nhất và dãy con có tổng bằng S ở trên gồm các phần tử là các gói kẹo thuộc phần tử thứ nhất. Phần thứ hai là các gói kẹo còn lại.

c. Bảng phương án

N/Sdiv2	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
2	1	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0
3	1	0	1	0	1	0	1	0	1	0	1	0	1	0	0	0
4	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
5	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0

Dựa vào bảng phương án ta nhận thấy rằng số 14 là số lớn nhất thỏa :

- 14 ≤ T div 2 =15
- Có một dãy con của dãy a có tổng bằng 14 (F[5,14]=1).

d. Truy vết

Nếu bài toán yêu cầu truy vết thì ta làm tương tự bài trên nhưng bắt đầu tại vị trí F[5,14].

e. Cài đặt

```
Program chiakeo;
Const
        limit=10000;
Var
        A:Array[1..limit] of longint;
        F:Array[0..limit,0..limit] of byte;
        n,i:integer;
        s,t,kq:longint;
        fi,fo:text;
{-----------------------------------------------------------------------------------------}
Procedure inputdata;
        Begin
                assign(fi,'chiakeo.inp');
                reset(fi);
                readln(fi,n);
                for i:=1 to n do
                        begin
                                read(fi,a[i]);
                                s:=s+a[i];
                        end;
                close(fi);
        End;
{-----------------------------------------------------------------------------------------}
Procedure outputdata;
        Begin
                assign(fo,'chiakeo.out');
                rewrite(fo);
                write(fo,s-2*kq);
                close(fo);
        End;
{-----------------------------------------------------------------------------------------}
Procedure process;
        Begin
                for i:=0 to n do F[i,0]:=1;
                for i:=1 to n do
                        for t:=1 to (s div 2) do
                                if F[i-1,t]=1 then F[i,t]:=1
                                else if t-a[i]>=0 then if F[i-1,t-a[i]]=1
```

```
                                        then F[i,t]:=1;
            for t:=(s div 2) downto 0 do
                if F[n,t]=1 then    begin
                                        kq:=t;
                                        break;
                                    end;
        End;
{-----------------------------------------------------------------------------------------}
Begin
        inputdata;
        process;
        outputdata;
End.
```

Bài toán 2: Market (olympic BalKan 2000)

a. Phát biểu bài toán

Người đánh cá Clement bắt được n con cá, khối lượng mỗi con là ai, đem bán ngoài chợ. Ở chợ cá, người ta không mua cá theo từng con mà mua theo một lượng nào đó. Chẳng hạn 4kg, 6kg…

Ví dụ: có 3 con cá, khối lượng lần lượt là: 3, 2, 4. Mua lượng 6 kg sẽ phải lấy con cá thứ 2 và và thứ 3. Mua lượng 3 kg thì lấy con thứ nhất. Không thể mua lượng 8 kg.

Nếu bạn là người đầu tiên mua cá, có bao nhiêu lượng bạn có thể chọn?

Inputdata: file văn bản **market.inp**

- Dòng 1 : 1 số nguyên dương N duy nhất ($n \le 10^5$)
- Dòng 2 : N số tiếp theo cách nhau bởi 1 dấu cách chỉ khối lượng của N con cá

Outputdata: file văn bản market.out: Cho biết tổng lượng bạn có thể mua

Market.inp	Market.out
3 2 3 4	7

b. Hướng dẫn giải

Thực chất bài toán là tìm các số S mà có một dãy con của dãy a có tổng bằng S. Ta chỉ cần đếm xem trong bảng phương án có bao nhiêu F[n,t]=1(t=1..S) thì có bấy nhiêu phương án.

c. Cài đặt

```
Program market;
Const
        limit=10000;
Var
        A:Array[1..limit] of longint;
        F:Array[0..limit,0..limit] of byte;
        n,i:integer;
```

```
        s,t,kq:longint;
        fi,fo:text;
{-----------------------------------------------------------------------------------------}
Procedure inputdata;
        Begin
                assign(fi,'market.inp');
                reset(fi);
                readln(fi,n);
                for i:=1 to n do
                        begin
                                read(fi,a[i]);
                                s:=s+a[i];
                        end;
                close(fi);
        End;
{-----------------------------------------------------------------------------------------}
Procedure outputdata;
        Begin
                assign(fo,'market.out');
                rewrite(fo);
                write(fo,kq);
                close(fo);
        End;
{-----------------------------------------------------------------------------------------}
Procedure xuly;
        Begin
                for i:=0 to n do F[i,0]:=1;
                for i:=1 to n do
                        for t:=1 to s do
                                if F[i-1,t]=1 then F[i,t]:=1
                                else if t-a[i]>=0 then if F[i-1,t-a[i]]=1
                                                        then F[i,t]:=1;
                for t:=1 to s do
                        if F[n,t]=1 then kq:=kq+1;
        End;
{-----------------------------------------------------------------------------------------}
Begin
        inputdata;
        xuly;
        outputdata;
End.
```

Bài toán 3: Điền dấu

a. Phát biểu bài toán

Cho n số tự nhiên $a_1, a_2, ..., a_n$. Ban đầu các số được đặt liên tiếp theo đúng thứ tự cách nhau bởi dấu "*": $a_1*a_2*...*a_3$. Cho trước số nguyên S, có cách nào thay các dấu "*" bằng dấu '+' hay dấu '− 'để được một biểu thức số học cho giá trị là S không?

Yêu cầu : Tìm cách thay thế thỏa mãn bài toán.

Inputdata: file văn bản **diendau.inp**

- Dòng đầu tiên gồm hai số nguyên dương N (số các số tự nhiên) và S. ($1 \le N \le 10^3$; $1 \le S \le 10^5$).
- Dòng tiếp theo gồm N số là các số tự nhiên ($0 \le a_i \le 10^2$).

Outputdata: file văn bản **dieudau.out**: Yes nếu như có cách thay thế, No nếu không có cách thay thế.

Example:

Diendau.inp	Diendau.out
9 5 1 2 3 4 5 6 7 8 9	Yes

Giải thích : 1-2+3-4+5-6+7-8+9=5.

b. Hướng dẫn giải

Đặt F[i,t] =1 nếu có thể điền dấu vào i số đầu tiên và kết quả bằng t. Ta có thể tính F theo công thức sau.

- F[1,a[1]]=1
- F[i,t]=1 nếu F[i-1,t+a[i]]=1 hoặc F[i-1,t-a[i]]=1

Nếu F[n,S]=1 thì câu trả lời của bài toán là có.

Chú ý rằng chỉ số t theo mảng phải có cả phần âm (tức là từ -T đến T, với T là tổng của n số). Vì trong bài này chúng ta dùng cả dấu trừ nên có thể tạo ra các tổng âm.

c. Bảng phương án

N/t	*-7*	*-6*	*-5*	*-4*	*-3*	*-2*	*-1*	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
2	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0
3	0	0	0	1	0	0	0	1	0	0	0	0	0	1	0
4	0	0	1	0	0	0	1	0	1	0	0	0	1	0	1

d. Cài đặt

Program DienDau;

```
Const
        limit = 100000;
Var
        F: array[0..1000,-limit..limit] of byte;
        A: array[1..1000] of integer;
        n,i:integer;
        S,t,max:longint;
        fi,fo:text;
{-----------------------------------------------------------------------------------}
Procedure inputdata;
        Begin
                assign(fi,'diendau.inp');
                reset(fi);
                readln(fi,n,s);
                for i:=1 to n do
                        begin
                                read(fi,a[i]);
                                max:=max+a[i];
                        end;
            close(fi);
        End;
{-----------------------------------------------------------------------------------}
Procedure outputdata;
        Begin
                assign(fo,'diendau.out');
                rewrite(fo);
                if F[n,s] = 1 then writeln(fo,'yes')
            else writeln(fo,'no');
            close(fo);
        End;
{-----------------------------------------------------------------------------------}
Procedure xuly;
        Begin
                F[1,a[1]]:=1;
                for i:=2 to n do
                        for t:=-max to max do
                                if (F[i-1,t+a[i]]=1) then
                                        F[i,t]:=1
                        else if t-a[i]>=0 then if (F[i-1,t-a[i]]=1)
                                                then F[i,t]:=1;
      End;
{----------------------------------------------------------------------------------}
Begin
        inputdata;
        xuly;
        outputdata;
```

End.

Nhận xét: Lớp bài toán này chúng ta có thể liệt kê vào lớp bài toán cái túi.

BÀI 7: LỚP BÀI TOÁN NHÂN MA TRẬN

Tiền Đề: Cách nhân 2 ma trận

Example:

$$\begin{bmatrix} 1 & 0 & 2 \\ -1 & 3 & 1 \end{bmatrix} \times \begin{bmatrix} 3 & 1 \\ 2 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} (1\times 3 - 0\times 2 + 2\times 1) & (1\times 1 - 0\times 1 + 2\times 0) \\ (-1\times 3 + 3\times 2 + 1\times 1) & (-1\times 1 + 3\times 1 + 1\times 0) \end{bmatrix}$$

Thuật toán:

```
nếu columns[A] ≠ rows [B]
then eror
else for i ←1 to rows[A]
    do for j ←1 to columns [B]
        do C[i,j] ← 0
            for k ← 1 to columns [A]
                do C[i,j] = C[i,j] + A[i,k].B[k,j]
Return C
```

Lưu ý: Ta chỉ có thể nhân hai ma trận A và B nếu số lượng cột của A bằng số lượng hàng hàng của B. Nếu A là một ma trặng p x q và B là một ma trận q x r, ma trận kết quả C là một ma trận p x r. Tổng số phép nhân vô hướng để tạo ra ma trận C là pqr.

Tính chất của phép nhân ma trận:

- $(AB)C = A(BC)$ (kết hợp).
- $(A + B)C = AC + BC$ (phân phối bên phải).
- $C(A + B) = CA + CB$ (phân phối bên trái)

Để giảm số phép nhân chúng ta sẽ tránh tạo ra các ma trận trung gian có kích thước lớn và do phép nhân ma trận có tính kết hợp nên có thể đạt được điều này bằng cách sử dụng các dấu đóng mở ngoặc để chỉ ra thứ tự thực hiện các phép nhân ma trận.

I. Tổng quan

1. Mô hình

Cho 1 dãy ma trận $<A_1, A_2, \ldots, A_n>$. Ta muốn tính tích $A_1A_2\ldots A_n$, hãy xác định trình tự nhân (cách đặt các dấu ngoặc đơn sao cho hợp lý) để cho số phép nhân cần thực hiện là ít nhất.

Example: Giả sử chúng ta có 3 ma trận A_1, A_2, A_3 với kích thước tương ứng 10 x 100, 100 x 5 và 5 x 50. Số giải pháp sử dụng các dấu ngoặc là 2:

- $((A_1A_2)A_3)$ = 10.100.5 + 10.5.50 = 7500
- $(A_1(A_2A_3)$ = 100.5.50 + 10.100.50 = 75000

Như vậy tính toán theo phép ngoặc đơn đầu tiên nhanh hơn gấp 10 lần.

Inputdata: file văn bản **nhanmatran.inp**:

- Dòng 1: Số nguyên N (N ≤ 100) cho biết số lượng ma trận
- N dòng tiếp theo: Dòng i ghi số dòng và số cột của ma trận thứ i (≤100)

Chú ý: Số dòng của ma trận sau phải bằng số cột của ma trận trước.

Outputdata: file văn bản **nhanmatran.out**: Ghi số lượng phép nhân ít nhất

Example

Nhanmatran.inp	Nhanmatran.out
3 10 100 100 5 5 50	7500

2. Hướng dẫn giải

Ta có nhận xét như sau: Một phép ngoặc đơn tối ưu của tích $A_1A_2\ldots A_n$ sẽ tách tích giữa A_k và A_{k+1} (1≤ k≤ n : k là vị trí đặt dấu ngoặc cho số phép nhân ít nhất) $(A_{1..k})(A_{k+1..n})$. Nghĩa là với một giá trị nào đó của k, trước tiên ta tính toán các ma trận $A_{1..k}$ và $A_{k+1..n}$ rồi nhân chúng với nhau để được tích cuối cùng $A_{1..n}$.

Gọi F[i,j] là số phép nhân ít nhất để tính tích ma trận $A_{i..j}$ (từ ma trận A_i đến ma trận A_j, trong đó 1≤ i ≤ j ≤ n). Như vậy F[1,n] là số phép nhân ít nhất để tích tính ma trận $A_{1..n}$.

Ta nhận thấy rằng:

- Nếu i=j thì dãy chỉ có một ma trận $A_{i..i}=A_i$, như vậy không cần phép nhân nào để tính tích các ma trận. Do đó **F[i,i]:=0**.
- Nếu i < j thì **F[i,j]:= F[i,k]+F[k+1,j]+$d_ic_kc_j$**

Chú giải: d_i số dòng của ma trận i, c_k số cột của ma trận k, c_j số cột của ma trận j.

K là một số nằm trong phạm vi từ i đến j. Bởi vậy ta cần phải kiểm tra tất cả các giá trị từ i đến j để tìm ra vị trí của số k. Do vậy chúng ta có công thức truy hồi như sau:

- **F[i,j]:=0** nếu i=j
- **F[i,j]:= min(F[i,k]+F[k+1,j)+ $d_ic_kc_j$)** nếu i < j

3. Bảng phương án

i/j	**1**	**2**	**3**
1	0	5000	7500

2	0	0	25000
3	0	0	0

4. Cài đặt

```
Program NhanMaTran;
Const
        limit=1000;
Type kichthuoc=record
        d: word;
        c: word;
        End;
Var
        F: Array[0..limit,0..limit] of longint;
        P: Array[1..limit] of kichthuoc;
        i,j,k,n,l,t: longint;
        fi,fo:text;
{-------------------------------------------------------}
Procedure inputdata;
        Begin
                assign(fi,'nhanmatran.inp');
                reset(fi);
                readln(fi,n);
                for i:=1 to n do readln(fi,p[i].d,p[i].c);
                close(fi);
        End;
{-------------------------------------------------------}
Procedure outputdata;
        Begin
                assign(fo,'nhanmatran.out');
                rewrite(fo);
            For i:=1 to n do
                  Begin
                        for j:=1 to n do write(fo,f[i,j],' ');
                        writeln(fo);
                  End;
            writeln(fo);
                write(fo,f[1,n]);
                close(fo);
        End;
```

```
{-----------------------------------------------------}
Procedure Process;
          Begin
                    for i:=1 to n do f[i,i]:=0;
          for l:=2 to n do
              for i:=1 to n-l+1 do
                  Begin
                      j:=i+l-1;
                     f[i,j]:=100000000;{vô cùng}
                     for k:=i to j-1 do
                         Begin
                            t:=f[i,k]+f[k+1,j]+p[i].d*p[k].c*p[j].c;
                            if t<f[i,j] then f[i,j]:=t;
                         End;
                  End;
          End;
{-----------------------------------------------------}
Begin
          inputdata;
          process;
          outputdata;
End.
```

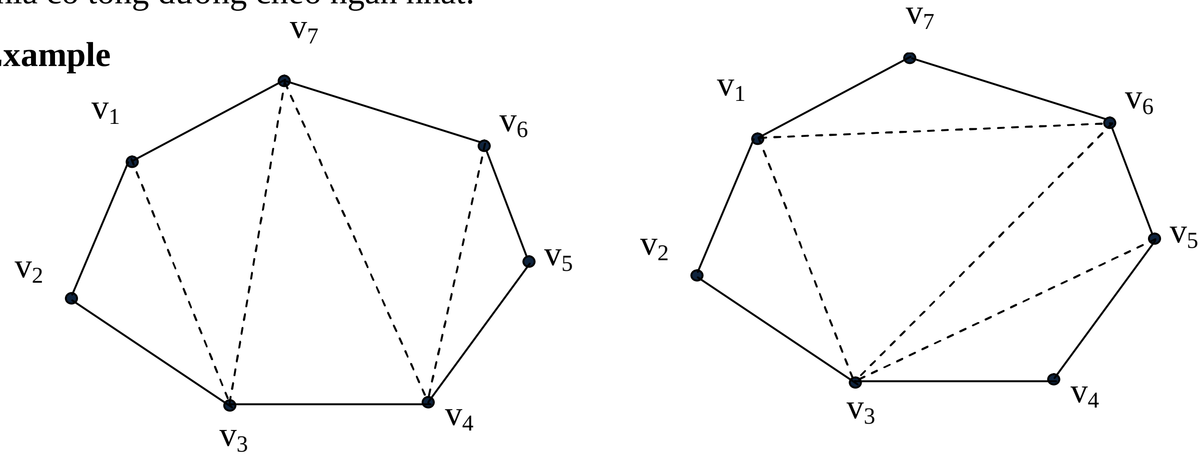
II. Áp dụng

Bài toán: Tam giác phân đa giác

a. Phát biểu bài toán

Cho P=<$v_1,v_2,\ldots,v_{n-1}$> là một đa giác lồi có N đỉnh và N cạnh ($v_1v_2,v_2v_3\ldots,v_nv_1$). Bằng các đường chéo không cắt nhau ta có thể phân đa giác thành N-2 tam giác. Hãy xác định cách chia có tổng đường chéo ngắn nhất.

Example



***Hình**: Hai cách phân một đa giác lồi. Mọi phép phân tam giác của đa giác có 7 cạnh luôn có 7-3=4 đường chéo và chia đa giác thành 7-2=5 tam giác.*

Inputdata: file văn bản **phandagiac.inp** gồm:

- Dòng 1: Số nguyên M (M≤1000) số đường chéo của 1 đa giác.
- M dòng tiếp theo: Mỗi dòng gồm 3 số a, b, c. a,b là hai đỉnh không kề nhau, c là khoảng cách giữa hai đỉnh a, b (đường chéo). (a,b,c≤100)

Outputdata: file văn bản **phandagiac.out**: Lưu tổng đường chéo ngắn nhất.

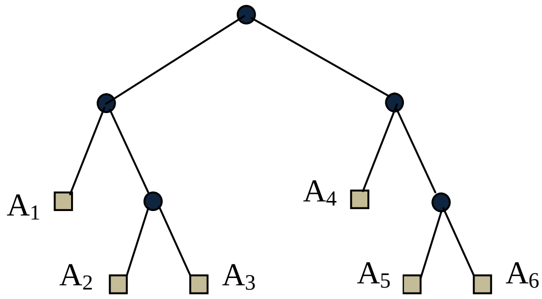
Example

Phandagiac.inp	Phandagiac.out
14 1 3 5 1 4 7 1 5 10 1 6 6 2 4 6 2 5 12 2 6 13 2 7 9 3 5 7 3 6 9 3 7 10 4 6 5 4 7 11 5 7 9	27

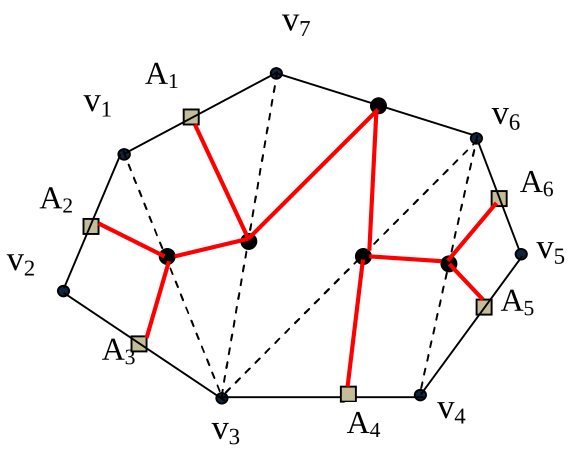
b. Hướng dẫn giải

Giả sử ta có tích 6 ma trận được biểu diễn như sau: $((A_1(A_2A_3))(A_4A_5A_6)))$

Ta có thể biểu diễn phép nhân 6 ma trận ở trên dưới dạng cây như sau:



Ta cũng có thể biểu diễn “phép tam giác phân đa giác” dưới dạng cây như sau:



Ta nhận thấy rằng bài toán “tam giác phân đa giác” chính là bài toán tính tích ma trận ở trên.

Gọi F[i,j] là tổng độ dài các đường chéo khi chia đa giác gồm các đỉnh từ i đến j thành các tam giác (j≥i+3: Vì đa giác phải có 4 đỉnh trở lên). Vậy ta có công thức truy hồi như sau:

- F[i,j]=0 với (j<i+3)
- F[i,j]= min(F[i,k]+F[k,j]+d[i,k]+d[k,j]): d[i,k] là độ dài đường chéo (i,j) (k=i+1..j-1).

c. Cài đặt: Các bạn tự cài đặt.

-----------------HẾT-----------------